DOCUMENT RESUME

ED 183 199 IR 008 142

AUTHOR Cox, John
TITLE Documentation--INFO: A Small Computer Data Base Management System for School Applications. The Illinois Series on Educational Application of Computers, No. 24e.
INSTITUTION Illinois Univ., Urbana. Dept. of Secondary Education.
SPONS AGENCY EXXON Education Foundation, New York, N.Y.
PUB DATE 79
NOTE 109p.: Best copy available. For related documents, see IR 008 123-140, and IR 008 144.

EDRS PRICE MF01 Plus Postage. PC Not Available from EDRS.
DESCRIPTORS *Computer Programs: Guides: Information Retrieval; Information Storage: Information Systems; *Management Information Systems; *Program Descriptions; Programers; Programing; Programing Languages; *Recordkeeping; *Schools
IDENTIFIERS *BASIC Programing Language; *Microcomputers

ABSTRACT

This paper documents the program used in the application of the INFO system for data storage and retrieval in schools, from the viewpoints of both the unsophisticated user and the experienced programmer interested in using the INFO system or modifying it for use within an existing school's computer system. The opening user's guide presents simple instructions with examples for using the INFO system to store data and search for certain entries. The programmer's guide covers size of files and necessary memory to use the programs, as well as the time it takes to run the programs; and a conversion guideline is included to explain those BASIC commands which are not always available on other microcomputer systems. A section on each program presents the program listing, the function of each variable, and suggestions for modifications which might be desirable for other applications. (Author/CMV)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

U.S. DEPARTMENT OF HEALTH, EDUCATION & WELFARE
NATIONAL INSTITUTE OF EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL NATIONAL INSTITUTE OF EDUCATION POSITION OR POLICY

Number 24e

ED183199

The Illinois Series on Educational Application of Computers

Documentation--INFO: A Small Computer Data Base Management System for School Applications

John Cox

College of Education
Department of Secondary Education
University of Illinois
Urbana, Illinois

"PERMISSION TO REPRODUCE THIS MATERIAL IN MICROFICHE ONLY HAS BEEN GRANTED BY

J. Richard Dennis

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

This work was supported by a grant from
The EXXON Education Foundation

Copyright, 1979, Department of Secondary Education
University of Illinois

IR008142

To the reader,

This is one of a series of documents prepared as resources for the preservice and inservice training of teachers. The general theme of all items in this series is "teaching with or about computers". The booklets are the collective work of a variety of staff and students (both present and former) of the College of Education, University of Illinois at Urbana-Champaign. The project generously was supported between 1977 and 1979 by a grant from the EXXON Education Foundation. All responsibility for the content of these documents rests with the authors and does not reflect necessarily the views of the EXXON Foundation. It was my extreme pleasure to direct this effort throughout.

J. Richard Dennis, PhD
Department of Secondary Education
University of Illinois

TABLE OF CONTENTS

Topic	Page
Introduction	1
USER'S GUIDE	3
Creating a Data File	4
Setup	4
Info	5
Notes on Info	12
Searching the File	14
Searc	15
Psear	19
World	21
Pull	22
Split	24
Expan	26
Comb	27
Chang	28
PROGRAMMER'S GUIDE	32
File Structure	32
Time Considerations	33
Memory Size	34
Conversion (Transportability) Guidelines	34
Program Listings and Variables	38
Bibliography	81

INTRODUCTION

One of the most useful applications of computers in the public schools is that of data storage and retrieval. Many records must be kept including administrative data and inventories of classroom materials. One system has been developed by the Secondary Education Department at the University of Illinois for keeping records on all graduate students. However, the computer programs used by the system are general enough to be used in many types of record-keeping applications. This paper documents the programs used in this application, called the INFO system, both from the view of the user (probably a secretary or faculty member not necessarily well versed in computer programming) and from the view of the programmer interested in using the INFO system or modifying it for use with his/her school's own computer system.

These programs were originally written for use with a DEC-10 timesharing computer (see ISEAC 9 and 13 for original documentation) and have been modified for use with the Micro-Computer System for Educational Applications (MSEA) now used by the Secondary Education Department (see ISEAC 20 and 27 for a description of the MSEA). Briefly, the MSEA is based on an 8080 micro-computer and floppy disk drives with sequential access files only. Also it is assumed in this documentation that there are two disk drives designated drive 0 and drive 1. The MSEA has several drawbacks which require the user to become somewhat familiar with file handling commands of the system (data files are stored on floppy diskettes). The efficiency of the INFO system is reduced as the data file becomes large (over 1000 entries) since it is necessary for the data file to be read sequentially (alphabetically) which may require several minutes.

Once data has been stored, the user may wish to search for entries which satisfy some special criteria. The computer can rapidly scan the data to determine which entries qualify and print a list of these entries. Here is an advantage in having the data computerized since performing such searches by hand is time-consuming and, in scanning larger data files, some entries may be left out which satisfy the search criteria. Once this material has been found, the user can statistically evaluate it to determine possible trends.

As data files become too large for use with INFO, the user may decide to split the file into smaller files which can be manipulated more rapidly. This feature now is included in the INFO system, as is the reverse situation of merging two small files to create a larger data file. The various programs and the functions they perform are shown below.

INFO System Programs and Functions

Program	Description
SETUP	This program is run only once for each data file. It permits the user to specify what kind of data will be kept in the file (referred to as headings or categories).
INFO	This program permits the user to enter, alter, delete or examine data.
SEARC	This program permits the user to ask the computer to search for items in the data file which satisfy various conditions.
PSEAR	Same as SEARC, but it permits the user to get a copy of the results of the search output to a printer.
WORLD	This program permits multiple searches for "arrays" of information and figures totals/averages for particular heading combinations.
SPLIT	Permits a data file to be "split" into two smaller files. The user also has the option of keeping the old data file intact while creating a duplicate or new file with some of the original data.
COMB	This program permits the user to merge two data files into one larger file. The only restriction is that both files have the same title for the first category.
CHANG	Permits the user to add, delete, or change the headings in a data file already being used. The new headings can be left blank or the user can specify some information to enter under that heading for each previous entry.
PULL	This program permits the user to create a new data file by searching an existing file for those entries which have some common characteristic. The user can specify which categories of information are to be put in the new file and whether the qualifying entries are to be removed from the old data file or left intact.
EXPAN	This program permits all the data stored in one data file to be copied into a blank file when the former file has been filled to capacity.

The programmer may wish to modify these programs for a different computer system or to create different features (such as printing requisition forms from inventories). A guide is provided to the logic of the programs and directions for making some modifications. Listings of the source programs (in BASIC) are presented and a brief description of some commands which are used in this version of BASIC, but are not available in other versions, will aid in transporting the INFO system to other computer configurations.

USER'S GUIDE

The following discussion is designed to present instructions which are simple enough so that even a person not accustomed to using computers will be able to use the INFO system for storing data and searching for certain entries. Throughout the following, an example will be used to illustrate the steps necessary to setting up and using a data file. While the main use of the data handling capabilities will probably be administrative in nature, this example focuses on use by teachers and students in a science classroom. (See ISEAC 17e for a discussion of other computer applications in teaching science.)

A science teacher decides to set up a data system to help keep track of inventory, to direct students to the location of supplies in the storeroom, and to provide students with possible hazards of using the various supplies in the course of laboratories. The first step in developing such a system is for the teacher to decide exactly what types of information will be stored in the computer. In this example, the following information has been chosen:

1) Name of Chemical
2) Alternate Name
3) Formula
4) Form or Concentration
5) Shelf Location
6) Possible Hazards
7) Quantity on Hand
8) Annual Needs
9) Catalog Number
10) Cost.

The first six categories are for use by the student or instructor in locating the exact chemical needed for experiments together with its safety information. The name of chemical information will be the common name and the data file will be alphabetized according to this heading (INFO always uses the first heading to order the data). Alternate names will be provided for organic chemicals and the concentration of solutions or form (crystal, powder, etc.) will be given where appropriate. The last

four categories will be helpful to the teacher in preparing requisitions to replace chemicals as the year progresses and may even eliminate the necessity for future inventories if the data in category 7 is updated after each experiment and when new supplies are received.

Creating a Data File

Once this information is determined, the teacher is ready to use the INFO system to create a data file and to store the information. A diskette is prepared with the INFO programs on it (referred to as the program disk) and the teacher then must allocate storage space on this diskette for the data file. To do this, create a file of about 50 sectors (about 12,800 characters) which is named DATA. The present programs assume that this file is created on disk drive 1.

Two other files, SCR and SCR1 should be allocated on another diskette and should be at least as large as the data file. These files are needed for updating the data file when making additions or corrections (see section below on INFO).

SETUP--entering the categories under which data is stored

Once the user has determined how the data file is to be set up and the necessary files have been allocated as discussed above, the next step is to use the SETUP program to enter the headings or categories which will be used in organizing the file. There is no limit to the number of categories which can be used with INFO, but searches will be slower and some program changes may be necessary if more than 10 or 20 headings are entered. In this paper, it is assumed that the user knows how to turn the computer on, load the operating system, and initiate execution of the involved BASIC program. When the computer is running the SETUP program, the user will be asked the name of the data file which is being established, how many data headings or categories are to be included in the file, and the name of each heading or category. The sample run shown below shows the messages and responses which a teacher in our example would enter in setting up the data file. Underlined text is entered by the operator. Lower case letters are comments and non-underlined upper case letters are computer output.

```
ALWAYS PRESS THE RETURN KEY AFTER TYPING IN A RESPONSE!!!!!!
TYPE IN THE NAME OF THE FILE YOU WANT TO CREATE.  NOTE THAT
SPACE FOR THE FILE MUST HAVE BEEN ALLOCATED
AND THE FILE MUST BE ON THE DISKETTE IN DRIVE 1
? DATA
HOW MANY HEADINGS WILL BE IN YOUR FILE
? 10
WHAT DO YOU WANT TO CALL HEADING NUMBER 1
? NAME OF CHEMICAL
```

```
WHAT DO YOU WANT TO CALL HEADING NUMBER 2
? ALTERNATE NAME
WHAT DO YOU WANT TO CALL HEADING NUMBER 3
? FORMULA
WHAT DO YOU WANT TO CALL HEADING NUMBER 4
? FORM OR CONCENTRATION
WHAT DO YOU WANT TO CALL HEADING NUMBER 5
? SHELF LOCATION
WHAT DO YOU WANT TO CALL HEADING NUMBER 6
? POSSIBLE HAZARDS
WHAT DO YOU WANT TO CALL HEADING NUMBER 7
? QUANTITY ON HAND
WHAT DO YOU WANT TO CALL HEADING NUMBER 8
? ANNUAL NEEDS
WHAT DO YOU WANT TO CALL HEADING NUMBER 9
? CATALOG NUMBER
WHAT DO YOU WANT TO CALL HEADING NUMBER 10
? COST
FILE DATA IS READY TO BE USED WITH OTHER PROGRAMS
OK
```

The message OK indicates that the program is completed and the user is ready to continue with the next program. The user should continue by loading the next program which is to be run--in this case, INFO.

INFO--changing or adding to the data file (see note 1)

The next step in creating the data file is to enter the information to be stored. The INFO program permits the user to do this.

Once the user has the INFO program running, the computer will request the name of the data file (DATA) and remind the user that the diskette containing the data file should be in drive 1 and that the diskette containing the two scratch files (SCR and SCR1) should be in drive 0. If the files are all properly located, the following message is then displayed:

```
CHOICES: 1) LOOK AT DATA FILE, 2) ADD, 3) ALTER,
         4) DELETE, 5) STOP THE PROGRAM.
```

The user can choose any of these options and the program frequently will return to this branching point. No changes or updates are made to the actual data file until option 5 is selected at which time the data file is modified. Any changes or additions to the file are stored in the scratch files until this time. (see note 2)

In setting up the data file, the teacher in our example would select option 2 since the data must be entered into the file. The computer then responds with the message

```
NAME OF CHEMICAL?  (PRESS RETURN FOR NONE)
```

and the user would begin entering the data requested for each heading or category. (See the sample run under the ADD option below.)

After entering as many chemicals as desired, the teacher would press the return key when the computer requests another name. This returns the user back to the main branching point. Finally, the teacher would select option 5 to stop the program. The computer at that time alphabetizes the additions entered (if necessary) and stores them in the data file. (see note 3) This may take several minutes depending on how many items are added to the file.

Once the information has been stored in the data file, the teacher may want to look at the file or correct typing mistakes. The following sections discuss the use of each of the options in the branching message.

1) LOOK AT DATA FILE

When option 1 is selected, the user has the opportunity to examine data which is stored in the data file. The computer prints:

```
CHOICES: 1) LIST OF NAMES, 2) DATA ON ONE ENTRY,
         3) DATA ON ALL ENTRIES, 4) NONE OF THESE
```

The first choice prints all the names of entries (category 1) stored in the data file in alphabetical order. The second option permits the user to view the data on just one entry by typing in the name of the entry desired. (see note 4) The third choice prints on the system terminal all the information in the data file, giving the user a chance to stop the print-out at the end of each page, and the fourth choice returns the user to the main program to choose another option. A sample run showing each of the four choices is given below:

```
CHOICES: 1) LOOK AT DATA FILE, 2) ADD, 3) ALTER,
         4) DELETE, 5) STOP THE PROGRAM.
? 1
CHOICES: 1) LIST OF NAMES, 2) DATA ON ONE ENTRY,
         3) DATA ON ALL ENTRIES, 4) NONE OF THESE.
? 1

ACETIC ACID
ACETONE
```

```
ALUMINUM
ALUMINUM CHLORIDE
AMMONIUM HYDROXIDE
BARIUM CHLORIDE
BARIUM CHLORIDE
BARIUM NITRATE
CALCIUM
CALCIUM CARBONATE
 .
 .
 .
COPPER (II) NITRATE
PRESS THE RETURN KEY FOR MORE.  TYPE ANYTHING ELSE TO STOP.
?  (user presses the return key)
COPPER SULFATE
 .
 .
 .
ZINC NITRATE
CHOICES: 1) LIST OF NAMES, 2) DATA ON ONE ENTRY,
         3) DATA ON ALL ENTRIES, 4) NONE OF THESE.
?  2
NAME OF CHEMICAL?  (PRESS RETURN KEY FOR NONE)
?  HYDROCHLORIC ACID

NAME OF CHEMICAL=HYDROCHLORIC ACID
ALTERNATE NAME=HYDROGEN CHLORIDE
FORMULA=H CL
FORM OR CONCENTRATION=4 M LIQUID
SHELF LOCATION=ROW 2 MIDDLE
POSSIBLE HAZARDS=CAN CAUSE BURNS.
QUANTITY ON HAND =12 LB.
ANNUAL NEEDS=24 LB.
CATALOG NUMBER=C95-M71 P.  1398 6 LB.
COST=5.25
NAME OF CHEMICAL?  (PRESS RETURN KEY FOR NONE)
?  (user presses return key)
CHOICES: 1) LIST OF NAMES, 2) DATA ON ONE ENtry
         3) DATA ON ALL ENTRIES, 4) NONE OF THESE.
?  3

NAME OF CHEMICAL=ACETIC ACID
ALTERNATE NAME=
FORMULA=H C 2 H 4 O H
FORM OR CONCENTRATION=12 M LIQUID
 .
 .
 .
FORMULA=AL CL 3
PRESS THE RETURN KEY FOR MORE.  TYPE ANYTHING ELSE TO STOP.
?  (user presses return key)
FORM OR CONCENTRATION=POWDER SOLID
```

```
  .  (data continues)
  .
  .
COST=3.59 (cost of last chemical)
CHOICES: 1) LIST OF NAMES, 2) DATA ON ONE ENTRY,
         3) DATA ON ALL ENTRIES, 4) NONE OF THESE.
?  4
CHOICES: 1) LOOK AT DATA FILE, 2) ADD, 3) ALTER,
         4) DELETE, 5) STOP THE PROGRAM.
(user  has  returned to the main branching point to choose another
option.)
```

2) ADD (see note 5)

When option 2 is selected, the computer requests the name of a new entry to be added to the data file. The user can press the return key to return to the main branching point to choose a different option or type the name of a new entry to the data file. (see note 6) The computer will search the file to see if the entry already is present in the file and will tell you if it is. If the entry does not exist, the computer requests the information to be stored under each of the headings or categories. The user also gets a chance to proof-read what was entered before the data is stored on disk. Any typing mistakes should be corrected before answering "NO" to the query "DO YOU WANT TO CHANGE ANY OF THESE ITEMS?" A sample run is shown below:

```
CHOICES: 1) LOOK AT DATA FILE, 2) ADD, 3) ALTER,
         4) DELETE, 5) STOP THE PROGRAM.
?  2
NAME OF CHEMICAL?
SILVER NITRATE
ENTER YOUR DATA
ALTERNATE NAME=?  (user presses return key)
FORMULA=?  AG N O 2        (user made error)
FORM OR CONCENTRATION=?  GRANULAR SOLID
SHELF LOCATION=?ROW 1 LEFT
POSSIBLE HAZARDS=?AVOID CONTACT WITH SKIN-CAUSES STAINS
QUANTITY ON HAND=?12 OZ.
ANNUAL NEEDS=?24 OZ.
CATALOG NUMBER=?C700-B10 (4 OZ) C700-315 P.  1415 POUND $6.49
COST=?19.85
THIS IS YOUR LAST CHANCE TO CHECK
THE DATA YOU JUST INPUT
1 NAME OF CHEMICAL=SILVER NITRATE
2 ALTERNATE NAME=
3 FORMULA=AG N O 2
4 FORM OR CONCENTRATION=GRANULAR SOLID
  .
  .
  .
10 COST=19.85
```

```
DO YOU WANT TO CHANGE ANY OF THESE ITEMS?
? Y
TYPE THE NUMBER OF THE ITEM YOU WANT TO CHANGE.
? 3
ENTER THE CORRECTED INFORMATION FOR FORMULA
? AG N O 3
THIS IS YOUR LAST CHANCE TO CHECK
THE DATA YOU JUST INPUT
1 NAME OF CHEMICAL=SILVER NITRATE
2 ALTERNATE NAME=
3 FORMULA=AG N O 3
 .
 .
 .
10 COST=19.95
DO YOU WANT TO CHANGE ANY OF THESE ITEMS?
? N
NAME OF CHEMICAL?
SILVER      (user enters name of a chemical already in data file)
SILVER ALREADY EXISTS.  TRY AGAIN.
NAME OF CHEMICAL
?  (user presses return key)
CHOICES: 1) LOOK AT DATA FILE, 2) ADD, 3) ALTER,
         4) DELETE, 5) STOP THE PROGRAM.
(user has returned to the main branching point.)
```

3) ALTER--changing data already stored in the data file

When option 3 is selected, the user can make changes to an entry already stored in the data file. (see note 2) The computer first asks for the name of the entry to be changed. The user can press the return key to return to the main program or type the name of an entry which is in the data file. (see note 6) The computer will search the file for the entry and, if it does not exist, will inform the user. If the entry does exist, the computer prints:

```
CHOICES: 1) CHANGE ONE HEADING ENTRY, 2) CHANGE MORE THAN
         ONE HEADING ENTRY, 3) SEE EACH ENTRY AND CHANGE
         ANY YOU WANT.
```

If the user selects choice 1, the computer requests the number of the category or heading to be changed. The user can type 0 to see a list of the headings used in the data file. If the user wants to change the name of the entry (category 1), the computer will search the data file for the new changed name to be sure that another entry by that name does not already exist. (see note 7) If it already exists, the user is so informed and told to try again. Otherwise, the change is stored and the user is asked for the next entry to be changed.

If the user selects choice 2 (CHANGE MORE THAN ONE HEADING ENTRY), the computer responds as in choice 1, except that after each change the computer prints, "DO YOU WANT TO CHANGE ANOTHER ITEM? (YES OR NO)" and the changes are not stored on the diskette until the user types, "NO". The user can continue to change items in the current entry at will until finished. Changing the name again results in a search of the data file to assure that the new name does not already have an entry in the file. (see note 7)

If the user selects choice 3 (SEE EACH ENTRY AND CHANGE ANY YOU WANT), the computer will print each category or heading separately with the message "TYPE C TO CHANGE OR PRESS THE RETURN KEY". As the user views each data item, if the information is correct, the user should press the return key and the next item will be displayed. To change the item, the user should type "C" (followed by the return key). The computer will request the corrected information and continue to display the rest of the information for the current entry. After the last item has been displayed and checked, the information is stored in the scratch file. As in the above cases, the user can change the name of an entry, but this will result in a search of the data file to assure that the new name does not already exist. (see note 8) A sample run is shown below:

```
CHOICES: 1) LOOK AT DATA FILE, 2) ADD, 3) ALTER
         4) DELETE, 5) STOP THE PROGRAM.
? 3
WHICH NAME OF CHEMICAL DO YOU WANT TO CHANGE?
PRESS RETURN FOR NONE.
? SILVER NITRATE
SILVER NITRATE

CHOICES: 1) CHANGE ONE HEADING, 2) CHANGE MORE THAN
         ONE HEADING, 3) SEE EACH HEADING AND CHANGE ANY YOU WANT.
TYPE IN THE NUMBER OF THE CHOICE YOU WANT.
? 1 (user wants to change quantity on hand only)
WHICH HEADING NUMBER DO YOU WANT TO CHANGE?
TYPE 0 FOR A LIST OF HEADINGS.
? 7
ENTER THE CORRECTED INFORMATION FOR QUANTITY ON HAND
? 12 G
WHICH NAME OF CHEMICAL DO YOU WANT TO CHANGE?
PRESS RETURN FOR NONE
? ZINC
ZINC

CHOICES: 1) CHANGE ONE HEADING, 2) CHANGE MORE THAN
         ONE HEADING, 3) SEE EACH HEADING AND CHANGE ANY YOU WANT.
TYPE IN THE NUMBER OF THE CHOICE YOU WANT.
? 3 (user wants to check data)
EACH ITEM WILL BE DISPLAYED.  IF YOU WANT TO CHANGE
THAT ITEM PRESS THE LETTER C AFTER THE QUESTION MARK.
```

```
IF YOU WANT TO LEAVE THE ITEM ALONE, PRESS THE RETURN KEY.
NAME OF CHEMICAL=ZINC
TYPE C TO CHANGE OR PRESS THE RETURN KEY.
?  (user presses return)
ALTERNATE NAME=
TYPE C TO CHANGE OR PRESS THE RETURN KEY.
?  C (user wants to change alternate name)
TYPE IN THE CORRECTED INFORMATION
?  MOSSY
FORMULA=ZN
TYPE C TO CHANGE OR PRESS THE RETURN KEY.
?  (user presses return)
  .
  .
  .
COST=4.17
TYPE C TO CHANGE OR PRESS THE RETURN KEY.
?  (user presses return)
WHICH NAME OF CHEMICAL DO YOU WANT TO CHANGE?
PRESS RETURN FOR NONE.
?  (user presses return)
CHOICES: 1) LOOK AT DATA FILE, 2) ADD, 3) ALTER,
         4) DELETE, 5) STOP THE PROGRAM.
(user has returned to the main braching point to select another
option.)
```

4) DELETE--removing an entry from the file

When option 4 is selected, the user is permitted to delete entries already stored in the data file. The computer requests the name of the entry to be deleted. (see note 5) The user can press the return key to return to the main program for another option or type the name of an entry which is no longer needed in the data file. The computer will search for the name to be sure the entry exists and the user is notified if the entry is not found. If the entry does exist, the computer prints the name and asks if the user is sure that this entry is to be deleted. (You get one last chance to save the data!) To decide not to delete, the user types "NO" and the computer will ask for a different name to delete. If the user types "YES" the information is stored in the scratch file and the entry will be deleted when the data file is updated at the end of the program. A sample deletion is shown below:

```
CHOICES: 1) LOOK AT DATA FILE, 2) ADD, 3) ALTER,
         4) DELETE, 5) STOP THE PROGRAM.
?  4
WHICH ENTRY DO YOU WANT TO DELETE?  (PRESS RETURN FOR NONE)
?  PICRIC ACID
THIS WILL DESTROY ALL DATA ON PICRIC ACID
ARE YOU SURE YOU WANT TO DELETE THIS ENTRY?  (YES OR NO)
?  Y
```

WHICH ENTRY DO YOU WANT TO DELETE? (PRESS RETURN FOR NONE)
? (user presses return)
CHOICES: 1) LOOK AT DATA FILE, 2) ADD, 3) ALTER,
4) DELETE, 5) STOP THE PROGRAM.
(user has returned to the main branching point to select another option.)

5) STOP THE PROGRAM--updating the file

When option 5 is selected, the computer determines if an update to the data file is needed (were any items added, changed, or deleted?). If it is necessary, the computer then alphabetizes these changes (see note 9) and updates the data file. This may require some time and the computer should not be turned off before the message "INFO IS NOW FINISHED" is displayed on the terminal, since the data file may be destroyed by such an interruption. (see note 10) The message OK or READY at the end of the program assures the user that the program run is completed and the user can type RUN to use the INFO program again or may load a different program.

USER NOTES ON INFO

1. If the data file already has information stored in it, the user may need to run the program EXPAN before attempting to use the INFO program to make additions to the file. (See the section on EXPAN below.)

2. Changes to the data file are not completed until the user chooses option 5. Therefore, if you catch a mistake in an item you just finished adding and attempt to change it, the computer will not be able to find that entry in the data file since it has not actually been added to the file until option 5 is run. Any mistakes that are noticed in the newly entered data can not be corrected until the user chooses option 5, the file is updated, and the INFO program is run again.

3. Updating the data file is a time consuming process and the user is advised to make all changes possible before selecting option 5. Some time is also saved if the changes are made in alphabetical order, but this is not necessary if the computer time is available.

4. The name entered for searching need not be exact. The computer will compare what you type with the first letters of each entry and return the first match it finds. For example, if the data file contains entries under SODIUM ACETATE, SODIUM CHLORATE, and SODIUM CHLORIDE, the user could type SODIUM and the computer would return with SODIUM ACETATE. However, care should be taken in using this feature, since the user might type SODIUM C expecting to get SODIUM CHLORIDE, but would instead get SODIUM

CHLORATE since it is alphabetically the first match. We advise users to enter the whole name unless there is some doubt about the spelling or unless the user is positive that no other entry exists with the same beginning letters.

5. No commas are permitted in data that is entered using the INFO program. (This is due to the features of the BASIC language which reserves commas to separate inputs.) If a comma is entered accidentally, only the data before the comma will be accepted--the rest will be ignored. Another feature of BASIC on our computer is that only the first 72 characters entered under any category will be properly stored. Users should take care not to exceed this limit since it might cause all data in the file alphabetically greater than the long line to be destroyed!!!!! (See Programmer's Guide)

6. Special care should be used when entering a name to be deleted or added. (See also note 4.) If the user decides to change the name before storing it, the file will have to be searched again. If the wrong entry is returned in deleting, the user may accidentally destroy good data. Again, we urge users to type the whole name for an entry to avoid problems.

7. If the user presses the return key instead of typing a new name, the computer ignores the change and asks what category number you want to change. (This is because the system cannot accept a blank name for the first entry.) It will be necessary to actually type some data (either corrected or the same as already stored) before the computer will end this phase of correcting data.

8. While choice 2 also permits the user to scan any of the information stored under a specific entry, choice 3 is better if the user is not sure that changes are actually needed. In order to see information and not change it using choice 2, the user must retype each item viewed or lose the data stored under that category (see also note 7 above). Using choice 3, the user simply presses the return key if no change is desired to the item being displayed.

9. The program first checks to see if the changes were entered alphabetically in which case the update begins sooner. The time it takes to alphabetize the changes depends on how many changes were entered during the session. See the section on time considerations in the Programmer's Guide for more details on the actual time required.

10. Actually, there is no danger to the data file until the process of copying the update file back into the data file begins. Depending on the data file size and the number of changes made, this may not occur for several minutes. However, you are much

safer to leave the computer running until the end message is printed.

Searching the File for Specific Information

There are three separate programs for searching the data file for entries which satisfy certain user-specified criteria: SEARC, PSEAR, and WORLD. These programs can save the user much time and effort over manual searches of the data and are capable of conducting simple or more complex types of searches. SEARC and PSEAR are essentially the same program--SEARC permits the user to view the results of the search on the system terminal, while PSEAR uses the system's line printer to provide a written copy of the results which the user can carry away. WORLD provides other search types and will be discussed separately.

SEARC and PSEAR allow the user to specify headings or categories to be searched according to 6 different search strategies:

1) look for data under one of the headings or categories which matches a user-specified standard exactly (see note 1 below)

2) look for data under one heading or category which has the standard as the first few characters

3) look for data under one heading which is less than or equal to (alphabetically or numerically preceding) the standard (see note 2 below)

4) look for data under one heading which is greater than or equal to the standard

5) look for numeric data under one of the headings, count how many entries have such numeric data, total the numbers, and compute the average of the numbers (see notes 3 and 4 below)

6) look for data under one category that matches the standard exactly but may be embedded in the line of information (see note 5 below)

The entries which satisfy the search criterion are displayed on the system terminal. (If a printed copy of the search results is needed, the user should run PSEAR or hook the computer to a printing terminal.)

Up to 50 different search criteria can be specified for any search and the user can decide whether the search is to be a logical "and" or an "or". (An "and" search means that an entry must satisfy all of the search criteria before it is accepted and displayed. An "or" search means that an entry will be accepted if it satisfies any of the criteria.) (see note 6 below)

The user also can specify which categories or headings will be printed out or displayed on the terminal for those entries which satisfy the search criteria. Information which is not needed does not have to be shown.

SEARC

In order to use the SEARC program, the user should place the program diskette in drive 1 and follow the same procedures as used in loading the SETUP program.

For this program, the computer does not require scratch files. The computer will request the user to enter the name of the data file and to specify what kind of terminal is being used (paper-printing, such as a teletype, or cathode ray terminal, similar to a television display). If a paper-printing terminal is being used, the computer also will ask for the date so that the listing of search results can be documented. A help sequence can be requested which will remind the user of the features of the program. Also, the user may request a list of the headings or categories used in the data file as well as a list of the possible search types.

Next, the computer prints:

SPECIFY THE HEADING NUMBER(S) AND TYPE(S) OF SEARCHES THAT YOU WANT; ENTER 0 FOR HEADING NUMBER WHEN DONE.

Suppose the teacher in our example wanted a list of all the acids in stock and how much is left of each acid. The teacher could accomplish this by using a number 6 search for the word "acid" in the name field of each entry and requesting the computer to print the name and quantity on hand (categories 1 and 7) for each of the entries which satisfies this simple search criterion. (Of course, for a small file, the teacher merely could view the data and note the amounts of each acid, but as the file becomes larger, the job of visually inspecting it becomes more time-consuming.)

The teacher would type "1" in response to the query for HEADING NUMBER from the computer since we want to search the name category of each chemical for the word "acid". The computer then prints: SEARCH TYPE NUMBER and the teacher types in "6". Next, the computer prints: STANDARD and the teacher types "ACID". The computer then asks for another heading number, but since this search is simple and no other criteria need be specified, the teacher would type "0".

In order to print only the name of each acid and the quantity on hand, the teacher must specify the headings which are to be displayed. The computer prints:

```
NOW SPECIFY THE NUMBERS OF ALL THE HEADINGS YOU WANT PRINTED.
ENTER 0 WHEN DONE OR -1 FOR ALL THE HEADINGS.
```

The teacher would type "1", press return, type "7". press return, and type "0".

The computer then prints:

```
ARE THE RESULTS TO BE LISTED IN COLUMN FORM (YES OR NO).
```

Column form means that each entry which satisfies the search criteria will be displayed on a single line (see note 7). The alternative method is for the information on each category requested to be printed on a separate line. The teacher in our example decides on column form since it is easier to read and types "yes". See the sample run below for the representation of this example and the results of the search.

```
WHAT IS THE NAME OF THE DATA FILE IN DRIVE 1
? DATA
ARE YOU USING A PAPER-PRINTING TERMINAL (YES OR NO) ? NO
DO YOU NEED HELP (YES OR NO) ? NO
DO YOU WANT A LIST OF HEADINGS (YES OR NO) ? NO
DO YOU NEED A LIST OF SEARCH TYPES (YES OR NO) ? NO
SPECIFY THE HEADING NUMBER(S) AND TYPE(S) OF SEARCHES
THAT YOU WANT; ENTER 0 FOR HEADING NUMBER WHEN DONE
HEADING NUMBER? 1
SEARCH TYPE NUMBER ? 6
STANDARD ? ACID
TYPE A HEADING NUMBER TO SPECIFY ANOTHER SEARCH CONDITION,
TYPE 0 IF YOU ARE DONE, OR -1 TO START ALL OVER.
HEADING NUMBER ? 0 (user is done specifying criteria)

NOW SPECIFY THE NUMBERS OF ALL THE HEADINGS YOU WANT
PRINTED.  ENTER 0 WHEN DONE, OR -1 FOR ALL THE HEADINGS.
? 1
? 7
? 0
ARE THE RESULTS TO BE LISTED IN COLUMN FORM (YES OR NO) ? YES
NAME OF CHEMICAL              QUANTITY ON HAND
ACETIC ACID                   12 LB.
HYDROCHLORIC ACID             14 LB.
 .
 .
 .
SULFURIC ACID (CONC.)         5 LB.

12 OF THE 203 ITEMS IN THE FILE MEET THE REQUIREMENTS.
```

```
DO YOU WANT TO PERFORM ANOTHER SEARCH (YES OR NO) ? NO
```

More Complicated Searches

The user is permitted to specify up to 50 different criteria for a search which may all be required for an entry to be selected or the user may require that an entry which satisfies at least one of the criteria is to be selected. To demonstrate how the user might combine these features, suppose that the teacher in our example wants to get a list of the chemicals which cost between $1 and $5 to replace and what the average cost is for these chemicals. The sample run below shows how this can be accomplished.

```
WHAT IS THE NAME OF THE DATA FILE IN DRIVE 1
? DATA
  .
  .
  .
SPECIFY THE HEADING NUMBER(S) AND TYPE(S) OF SEARCHES
THAT YOU WANT; ENTER 0 FOR HEADING NUMBER WHEN DONE
HEADING NUMBER? 10 (user specifies cost)
SEARCH TYPE NUMBER? 4 (greater than or equal to)
STANDARD? 1
TYPE A HEADING NUMBER TO SPECIFY ANOTHER SEARCH CONDITION,
TYPE 0 IF YOU ARE DONE, OR -1 TO START ALL OVER
HEADING NUMBER? 10
SEARCH TYPE NUMBER? 3 (less than or equal to)
STANDARD? 5
TYPE A HEADING NUMBER TO SPECIFY ANOTHER SEARCH CONDITION,
TYPE 0 IF YOU ARE DONE, OR -1 TO START ALL OVER
HEADING NUMBER? 10
SEARCH TYPE NUMBER? 5 (compute averages and totals)
TYPE A HEADING NUMBER TO SPECIFY ANOTHER SEARCH CONDITION,
TYPE 0 IF YOU ARE DONE, OR -1 TO START ALL OVER
HEADING NUMBER? 0 (user is finished)

NOW SPECIFY THE NUMBERS OF ALL THE HEADINGS YOU WANT
PRINTED.  ENTER 0 WHEN DONE, OR -1 FOR ALL THE HEADINGS.
? 1
? 10
? 0 (user wants only name and cost)
ARE THE RESULTS TO BE LISTED IN COLUMN FORM (YES OR NO)? YES
NAME OF CHEMICAL          COST
ALUMINUM                  2.70
ALUMINUM CHLORIDE         3.85
BARIUM CHLORIDE           2.09
  .
  .
  .
ZINC NITRATE              3.59
51 OF THE 203 ITEMS IN THE FILE MEET THE REQUIREMENTS.
```

TOTALS UNDER HEADING COST = 150.65
AVERAGE OF VALUES FOUND = 3.15

The types of searches can be combined to scan a large data file for entries which fit very narrow or very broad specifications. By use of the "any" or "all" options, the user can specify different criteria which meet specific needs. For example, the "all" search requires that every criterion specified must be met in order for an entry to be counted. By specifying several such criteria, the possible entries are narrowed down. The "any" search can be used for broad searches: an item which meets at least one of the criteria will be counted. The number feature permits users to get totals and averages for numeric data and every search will print how many entries satisfied the criteria. The SEARC program can, therefore, be very useful in data analysis if the file is set up with that purpose in mind.

Notes on SEARC

1. An exact match means precisely what it says: the information stored under the selected heading or category must be exactly the same as the standard. This includes spelling, spacing, and punctuation.

2. Order is determined by the ASCII coding of the characters used in the data. A chart is included in the Programmer's Guide showing these codes. In general, numbers are less (in value) than capital letters, which, in turn, are less than small letters. Punctuation is less than numbers as are blank spaces. Therefore,

"GOTO" > "GO TO"
"ONE" > "115"
"hi" > "Hi"

This precedence should be kept in mind when setting up the file and when conducting searches.

3. In order for data to be treated as a number, the only characters used under that heading must be digits, positive or negative signs, or a decimal point. Spaces are also permitted, but no other characters may appear (no dollar signs or units such as g for grams, etc.). Spaces may cause problems when used to separate digits (the number 12 5.0 is valid as 125, but if the user wrote 3 5 9 to represent the three distinct integers 3, 5, and 9, the computer would drop the spaces and assume the user meant the number 359). Users should take care in setting up the data file that those fields later to be searched for numbers indeed have only proper numbers in them.

4. It is recommended that, whenever a number 5 search is conducted, the user ask for a printout of all acceptable entries and include in this printout the category or categories which have been searched for numbers. Values may be missed or misinterpreted.

5. The user should be careful to specify a standard which will find only the desired entries--the computer does not take order into account for a type 5 search. If the user specifies a standard of "CH", those entries which have the letters "CH" anywhere in the name will be found. This would return all chemicals with the word "chloride" in them, but would also find "chromate", "tetrachloride", and "chlorate".

6. Using the number 5 search when more than one search criterion is specified may cause strange results. If a number is found on an ANY search, the entry will be accepted even though it may not satisfy any other criterion. When using an ALL search, the number 5 search should be the first criterion specified since the criteria are applied in order--if an entry fails some other criterion before the number 5 search, that entry's number will not be included in the total or average which is computed.

7. If the amount of requested data is longer than can be displayed on a single line, the user should choose not to see the data in column form. The columns will be truncated (some information left out) and some pertinent data may not be displayed if the lines are too long.

PSEAR--line printer output of search results

The search types and procedures for PSEAR are the same as those for SEARC. However, the output from PSEAR is formatted to print on a (high speed) printer rather than a slower system terminal. To load PSEAR, follow the same steps as those used to run SETUP.

The same messages and searches are used in PSEAR (see SEARC above for examples). One additional feature is the program's ability to format output for printing labels. The teacher in our example could put standard adhesive labels (purchased for the line printer) into the high speed printer and allow the computer to print one for each chemical (or for a few chemicals) and place these labels on containers for easy identification in the stockroom. These labels would contain only the printed information which is requested by the user and must be printed in row form (The program presently is set to print 4 blank lines between entries, but this can be changed for other size labels--see the Programmer's Guide). Part of a sample run and an example of the labels are shown below.

ALWAYS PRESS THE RETURN KEY AFTER TYPING IN A RESPONSE

```
WHAT IS THE NAME OF THE DATA FILE IN DRIVE 1
? DATA
WHAT IS TODAY'S DATE (NO COMMAS PLEASE) ? 06-30-79
DO YOU NEED HELP (YES OR NO) ? NO
DO YOU WANT A LIST OF HEADINGS (YES OR NO) ? NO
DO YOU NEED A LIST OF SEARCH TYPES (YES OR NO) ? NO
SPECIFY THE HEADING NUMBER(S) AND TYPE(S) OF SEARCHES
THAT YOU WANT; ENTER 0 FOR HEADING NUMBER WHEN DONE
HEADING NUMBER? 1
SEARCH TYPE NUMBER ? 4
STANDARD ? AAAAA (user wants all names in file)
TYPE A HEADING NUMBER TO SPECIFY ANOTHER SEARCH CONDITION
TYPE 0 IF YOU ARE DONE, OR -1 TO START ALL OVER
HEADING NUMBER ? 0

NOW SPECIFY THE NUMBERS OF ALL THE HEADINGS YOU WANT
PRINTED.  ENTER 0 WHEN DONE, OR -1 FOR ALL THE HEADINGS.
? 1
? 2
? 3
? 4
? 0 (user wants only first four categories)
ARE THE RESULTS TO BE LISTED IN COLUMN FORM (YES OR NO) ? NO
DO YOU WANT LABELS (YES OR NO) ? YES
ROLL PAPER TO THE TOP OF THE PAGE AND PRESS RETURN ? (user
presses return)
(labels are printed on the high speed printer)
DO YOU WANT TO PERFORM ANOTHER SEARCH (YES OR NO) ? NO
```

The labels which would be printed as a result of running this program would look like this:

```
ACETIC ACID

H C 2 H 4 O H
  M LIQUID



ALUMINUM CHLORIDE

AL CL 3
GRANULAR SOLID
```

One label would be printed for each chemical in the data file.

WORLD--another kind of search

The WORLD program is intended as a supplement and augmentation of the SEARC program. Its main purpose is to facilitate searching the data file in a cross-tabs fashion using the number 3 search type discussed in the section on SEARC. This search looks for a user-specified standard anywhere in a specific heading or category of each entry. The program is loaded using the same procedures as those discussed for SETUP.

The user specifies the headings to be searched (a maximum of 10), what standards to use (a maximum of 20 per heading), and what headings contain only numbers (if any) for which averages are to be calculated (a maximum of 5 headings can be averaged if they all contain only numbers). The program searches through the data file (which may be very time consuming) counting how many entries satisfy the various criteria and prints a cross-tab table and any desired totals and averages all on the system terminal. (No hard copy is printed unless the terminal itself is a printer or teletype.)

Suppose the teacher in our example wanted to figure the average cost of compounds in liquid form which contain a certain element (say, hydrogen or sodium) as compared to the cost of compounds in solid form. A sample run is shown below illustrating the procedures and results obtained from such a search.

```
ALWAYS PRESS RETURN AFTER TYPING A RESPONSE!

NAME OF DATA FILE
? DATA
TYPE HEADING NUMBER YOU WANT INCLUDED, OR 0 FOR LIST OF
HEADINGS, OR -1 TO START ALL OVER, OR 11 TO STOP
SPECIFYING HEADINGS.  YOU CAN SPECIFY A MAXIMUM OF 5
HEADING NUMBERS.
(note that the 11 is the total number of headings in the file plus
1--other data files may have a different number of headings, so
this number to terminate the list may vary accordingly)
? 3
TYPE STANDARD(S) TO BE LOOKED FOR UNDER HEADING 3 (FORMULA).
IF STANDARD PRESENT ANYWHERE IN FIELD, ITEM WILL BE COUNTED.
TYPE DONE WHEN FINISHED, OR OVER TO START OVER
? " H "  (include spaces around "H" so only hydrogen is found)
? NA
? DONE
NEXT HEADING
? 4
? SOLID
? LIQUID
? DONE
NEXT HEADING
? 11
```

```
TYPE HEADING NUMBER(S) YOU WANT TOTAL/AVERAGE FOR, OR TYPE
0 FOR A LIST OF HEADING NUMBERS, OR TYPE -1 TO START OVER, OR
TYPE 11 TO STOP SPECIFYING HEADER NUMBERS.
? 5
? 11

TOTALS (IN HEADING SPECIFIED ORDER)
H SOLID = 19
TOTAL UNDER HEADING 5 = 33.66 (N = 19).  AVERAGE VALUE = 1.87
H LIQUID = 11
TOTAL UNDER HEADING 5 = 34.76 (N = 11).  AVERAGE VALUE = 3.16
NA SOLID = 21
TOTAL UNDER HEADING 5 = 43.69 (N = 21).  AVERAGE VALUE = 2.08
NA LIQUID = 3
TOTAL UNDER HEADING 5 = 8.85 (N = 3).  AVERAGE VALUE = 2.95
```

This means the average cost of compounds containing sodium is 2.19 while for those compounds containing hydrogen it is 2.14. There are 22 compounds which contain hydrogen and 24 for those which contain sodium. The liquid and solids also are shown. This is a rather contrived example; however, for certain applications or other types of data files, the WORLD program can be very useful in statistical analysis.

PULL--creating sub-files

Once a data file has been created, the user may want to break it into smaller files which have some characteristic in common. The PULL program permits the user to specify search criteria similar to the number 5 search type discussed in SEARC and the results are output into a new file for use with the other INFO system programs. Space for the new file must be allocated before using the PULL program. The user may also have the data which is placed in the new file deleted from the old file if desirable.

Suppose the teacher in our example wishes to create a new file for storing information on the liquids in the stockroom (for example, to help students prepare solutions of different molarity). Before using the PULL program, the teacher would allocate space for the new file. To illustrate, we will call this new file DATA1. The PULL program is loaded following the same procedures as described for SETUP.

The computer asks for the names of the old and the new files and in which drives they are located. The user can see a list of heading numbers and names, if necessary, and is then requested to type in the numbers of the headings to be used in the new file (in our example, all headings are to be copied into the new file). Help can also be requested on how to set up search constraints. For this program the number 5 search is used (the user specifies

which heading or category and a standard, and the computer examines each entry in the data file to determine if the standard appears anywhere within the heading). Up to 20 different criteria can be specified in this program (all criteria must be satisfied before an entry will be copied into the new file). In addition, the user can specify whether the criterion is to be found in order for entries to be copied, or whether entries containing the standard are not to be included (this is accomplished by using 1 to represent those standards which must be found or 0 to indicate standards which, if found, exclude the entry from the new file). The teacher in our example wants all liquids, but not those containing mercury, so the searches are entered as follows:

4,LIQUID,1 (entries which contain the word "LIQUID" in heading 4 are to be included--a type 1 search)

3,HG,0 (entries with HG--Mercury--in the formula--heading 3--are not to be included--a type 0 search).

A sample run is shown below:

```
NAME OF FILE CONTAINING DATA ? DATA
IN WHICH DRIVE IS THE DISKETTE CONTAINING THE DATA (0 OR 1) ? 1
WHAT IS THE NAME OF THE FILE YOU ARE ABOUT TO CREATE ? DATA1
WHICH DRIVE HAS THE DISKETTE CONTAINING FILE DATA1 (0 OR 1) ? 1
TYPE IN THE HEADING NUMBERS FOR THE FIELDS YOU WANT
STORED IN FILE DATA1.  TYPE 0 FOR A LIST OF HEADINGS AND
HEADING NUMBERS, OR TYPE 11 TO STOP SPECIFYING HEADING NUMBERS.
(again, the 11 is the number of headings in the file plus 1, and
this number to terminate the list will vary according to the
file's set-up)
? 1
? 2
? 3
? 4
? 5
? 6
? 7
? 8
? 9
? 10
? 11
DO YOU KNOW HOW TO SET UP FILE CONSTRAINTS (Y OR N) ? Y
TYPE IN THE HEADING NUMBER, STANDARD, OPTION (0 OR 1)
TYPE IN 0,0,0 WHEN DONE (NOTE: YOU CAN SPECIFY A MAXIMUM OF 20
CONSTRAINTS.
? 4,LIQUID,1
? 3,HG,0
? 0,0,0
DO YOU WANT TO REMOVE THE DATA ITEMS YOU JUST COPIED FROM
FILE DATA (Y OR N) ? N
DO YOU WANT TO RUN THIS PROGRAM AGAIN (Y OR N) ? N
```

If the user answers yes to the query "DO YOU WANT TO REMOVE THE DATA ITEMS YOU JUST COPIED FROM FILE DATA (Y OR N)", the computer will delete all the entries which satisfied the search criteria and were subsequently placed into the new file (see note 1 below). The computer will request the user in this case to place a scratch file (see note 2 below) in the other drive (the one without the old data file) and will print: WHEN READY, PRESS G AND THEN RETURN. IF YOU CHANGED YOUR MIND, JUST PRESS RETURN. This gives the user a second chance to be sure the data is to be deleted. If not, the user should press only the return key. If the data is indeed to be deleted and the scratch file is ready, the user should first press G and then the return key to begin the deletion process.

NOTES ON PULL

1. When deleting information from the old file, the computer destroys all data on each entry which satisfied the search criteria (not just the data which was copied to the new file).

2. The scratch file must be at least as large as the old data file and must be named SCR.

SPLIT--another method of separating data or enlarging files

The SPLIT program permits the user to break up a large data file into two smaller files, to remove the latter part of a data file, or to copy a data file into a larger file (see also EXPAN). Again, the new file must first be allocated before using the SPLIT program. The user is asked whether the entire file is to be copied or to specify a standard in which case only names greater than the standard will be placed in the new file. The user also may have the computer delete all names stored in the new file or leave them in the old file (see note 1 below).

Suppose the teacher in our example decides that it takes too long to search the data file and decides to split it into two smaller files, one containing the first part of the alphabet and the other containing the latter part. The teacher first would allocate a new file, say DATA2, on the program diskette. The teacher then would load the SPLIT program following the same steps as described in loading SETUP. The teacher gives the file names requested by the computer and types a standard of "MZZZZ" so that all chemicals beginning with A-M will remain in the old data file while those chemicals beginning with N-Z will be copied into file DATA2. The computer then asks whether the data which was copied (N-Z) should be deleted from the old data file. In this example the teacher replies YES and the computer deletes it. A sample run is shown below:

WHAT IS THE NAME OF THE EXISTING DATA FILE (WHICH MUST BE
ON THE DISKETTE IN DRIVE 1) ? DATA

```
WHAT IS THE NAME OF THE DATA FILE YOU ARE ABOUT TO CREATE ? DATA2
ON WHICH DRIVE IS THE DISKETTE WITH FILE DATA2 (0 OR 1) ? 1
TYPE 1 TO COPY ENTIRE FILE, 2 TO COPY A PORTION OF THE FILE, OR 0
TO QUIT THE PROGRAM ENTIRELY AND NOT COPY ANYTHING.
? 2
TYPE IN A LAST NAME--THE NAME NEED NOT EXIST IN FILE DATA
THE DATA FOR ANY ENTRY WITH A NAME ALPHABETICALLY
GREATER THAN THE NAME YOU TYPE WILL BE PLACED IN FILE
DATA2.  (TYPE JUST THE RETURN KEY TO QUIT THE PROGRAM.)
? MZZZZ
SHOULD THE DATA YOU JUST STORED IN FILE DATA2
BE REMOVED FROM FILE DATA (Y OR N) ? Y
ALL FINISHED!!
OK
```

The SPLIT program also may be used to enlarge the size of an existing data file (instead of using the EXPAN program discussed below). If the teacher in our example had some new chemicals to add to the file DATA and wanted to add more space to it, the first step would be to allocate a new file, say DATA3 (see note 2 below). The teacher then would run the SPLIT program again as shown below (see also note 3 below).

```
WHAT IS THE NAME OF THE EXISTING DATA FILE (WHICH MUST BE
ON THE DISKETTE IN DRIVE 1)? DATA
WHAT IS THE NAME OF THE DATA FILE YOU ARE ABOUT TO CREATE ? DATA3
ON WHICH DRIVE IS THE DISKETTE WITH FILE DATA3 (0 OR 1) ? 1
TYPE 1 TO COPY ENTIRE FILE, 2 TO COPY A PORTION OF THE FILE,
OR 0 TO QUIT THE PROGRAM ENTIRELY AND NOT COPY ANYTHING.
? 1
ALL FINISHED!!
```

NOTES ON SPLIT

1. When deleting data from the old file, the user must be sure a scratch file is provided. This file must be on the diskette in drive 0, it must be at least as large as the old data file, and it must be named SCR.

2. When making a copy of the data file for use as a backup, the new file needs to be as large as the old file.

3. The EXPAN program can also be used to enlarge the physical size of a data file and the user may find it easier (see the section below on EXPAN).

EXPAN--increasing the physical size of a data file

The EXPAN program is another way in which the physical size of an existing data file can be increased. It permits the user to make a second copy of the data file (as a back-up in case the original diskette is damaged) by copying the entire file into a newly allocated space on the same diskette or a new one. This process is necessary also in case more entries must be added to the data file after all the originally allocated space is consumed. (Note that the SPLIT program also can be used for this purpose.)

Before using the EXPAN program to make a copy of the data file, the user must allocate space for the new file on the program diskette or another diskette. Suppose the teacher in our example wants to increase the number of sectors in the data file (see note 1). The first step is to allocate space for the new file, say DATA4. Running the EXPAN program will copy the old file into DATA4.

To use the EXPAN program, follow the same steps as used in loading SETUP.

The program requests the names of both files and on which drives the diskettes containing the files are located. If the files are properly located, the program will copy the data file automatically. If an error occurs, the user is notified of the problem. A sample run is shown below:

```
NAME OF FILE YOU WANT TO EXPAND ? DATA
WHICH DRIVE IS DATA ON (0 OR 1) ? 1
NAME OF LARGER FILE ? DATA4      (see note 2)
WHICH DRIVE IS DATA4 ON (0 OR 1) ? 1
ALL FINISHED
OK
```

The OK indicates the program is finished and the user can then delete the old data file (if no longer needed) and change the name of the new data file, if desirable. Of course, if the new file is a back-up copy, the user would leave the program diskette alone and put the new diskette in a safe place.

Notes on EXPAN

1. Each sector holds 256 bytes of data (see discussion under Programmer's Guide). The best plan is to allocate plenty of room for the data file to allow for future expansion.

2. If the user is only making a back-up copy, the size of the new file need be only as large as the existing file.

COMB--merging two data files into one larger file

If two data files both have the same name for the first category of information, the COMB program can be used to merge the files into one containing all the entries in both files arranged alphabetically (the original data files remain unchanged for later use or as back-up copies of the information).

One of the original files (referred to as the ADD-ON file) is assumed to be more current than data in the other (referred to as the OLDEST file). If both files contain an entry with the same name, the data from the ADD-ON file replaces the data from the OLDEST file when they are merged (see note 1).

Suppose the teacher in our example originally had one data file (DATA5) which contained information on the liquids in the stockroom and a second file (DATA6) which contained information on the solids. These two files could be merged using the COMB program as long as the first heading in each file was the same (NAME OF CHEMICAL). Before running the program, the teacher would allocate space for a new file, say DATA7.

The COMB program can be loaded and run by following the same procedures as used for SETUP. The user will be asked by the program to enter the names of the data files and in which drive the diskette containing each file is located. The program then will merge the two files, which can take a long time (see note 2). The computer should not be interrupted until the message DATA7 HAS BEEN CREATED appears on the system terminal (see note 3). A sample run is shown below:

```
NAME OF 'OLDEST' FILE ? DATA5
ON WHICH DRIVE IS THE DISKETTE WITH FILE DATA5 (0 OR 1) ? 1
NAME OF 'ADDON' FILE ? DATA6
ON WHICH DRIVE IS THE DISKETTE WITH FILE DATA6 (0 OR 1) ? 1
NAME OF NEW FILE TO BE CREATED ? DATA7
ON WHICH DRIVE IS THE DISKETTE WITH FILE DATA7 (0 OR 1) ? 1
THIS IS PROBABLY GOING TO TAKE A WHILE.  I SUGGEST YOU GO
DO SOMETHING ELSE AND COME BACK LATER TO SEE IF I'M DONE.
DATA7 HAS BEEN CREATED
OK
```

If the two files contain the exact (see note 4) same heading names, the data from both files is placed alphabetically in the new file. If the files contain some different heading names, then the new file will contain a heading under every entry in every possible category (those entries which are missing certain categories will have a blank line under those headings).

Notes on COMB

1. The COMB program offers the user another way of updating data files. When changes or additions are needed, the user can run the INFO program, storing the changes in a new file and then later run the COMB program to merge the two files. In this case the user must be sure to declare the old file as the OLDEST, and the file containing the changes as the ADD-ON file.

2. Since our version of BASIC must read the files several times to merge them, it may take several minutes or longer (for long files) to run COMB. It is suggested that the program be used only when there will be no need for the computer, perhaps even overnight. See the Programmer's Guide for more information.

3. If it is necessary to interrupt the computer during the merge process, no real damage will be done to the old files. The user will just have to run the program again at a later time, starting from the beginning of the process.

4. When the program checks the heading names, it compares them from left to right. Spaces and punctuation count in this comparison so that ALTERNATENAME does not match ALTERNATE NAME, nor does MR. COX match MR COX. In the new file those headings which were in the OLDEST file will be listed first, followed by those headings, if any, which only appeared in the ADD-ON file.

CHANG--changing the data categories in a file

The CHANG program permits the user to add, delete, or change heading names used in an existing data file. If a heading is deleted, all the information under that heading for each entry is lost. When adding or changing a heading name, the user has the option of entering the information to be stored entry by entry or the new category can be set to any value chosen for all entries. The only restriction is that the new heading can not be the first category of information nor can the first heading be deleted since all information in the file is arranged alphabetically by that category. However, the name of the first heading can be changed as long as the user does not try to change the data stored under it.

Suppose the teacher in our example decided to add another category, perhaps the boiling point, for all chemicals in the data file. To make sure students will see this information, the teacher decides to have the new category listed before the possible hazards heading. The CHANG program can accomplish this.

The CHANG program is loaded and run following the same procedures as used for SETUP. The user must enter the name of the data file and place a diskette with a file named SCR in drive 0 (SCR must be at least as large as the data file). (See note 1) The computer then prints:

CHOICES: 1) ADD A NEW HEADING, 2) CHANGE A HEADING NAME (AND/OR DATA UNDER HEADING, 3) DELETE A HEADING, 4) LEAVE THIS PROGRAM.

The first option creates a new heading under which data can be stored for each entry. The second option permits the user to change the name of a heading and the third option deletes a heading (and all data stored under that heading for each entry). The fourth option is used to stop the program when finished. The user must also tell the computer which heading to change or delete or before which heading to add a new category.

If adding or changing a heading, the user next will get a message asking what to do about the data for each entry under that heading. When adding, the options are:

1) ENTER DATA UNDER HEADING NOW, 2) SET IT BLANK, or 3) SET TO SOME VALUE WHICH YOU CHOOSE.

Option 1 permits the user to enter the new data for each entry, one at a time, and requires the user to update all entries at one sitting. Option 2 leaves the data field blank and the user can add the data at another time using the INFO program. Option 3 permits the user to enter the same data for each entry in the file (again, these can later be changed using the INFO program). A sample run is shown below:

```
ALWAYS PRESS THE RETURN KEY AFTER TYPING-IN A RESPONSE!!!!!
MAKE SURE A DISKETTE LABELLED 'SCRATCH IS IN DRIVE 0.
NAME OF THE DATA FILE
THE FILE MUST BE ON A DISKETTE IN DRIVE 1.
? DATA

CHOICES: 1) ADD A NEW HEADING
         2) CHANGE A HEADING NAME
            (AND/OR DATA UNDER HEADING)
         3) DELETE A HEADING
         4) LEAVE THIS PROGRAM
TYPE IN THE NUMBER OF YOUR CHOICE ? 1
TYPE 1 TO ENTER IN ALL THE DATA UNDER THE NEW HEADING NOW, OR
TYPE 2 TO HAVE ALL THE DATA UNDER THE HEADING SET BLANK, OR
TYPE 3 TO SET ALL THE DATA UNDER THE HEADING TO A VALUE YOU
CHOOSE. ? 2
HERE ARE THE CURRENT HEADING NAMES AND THEIR NUMBERS:
1) NAME OF CHEMICAL
2) ALTERNATE NAME
3) FORMULA
```

```
4) FORM OR CONCENTRATION
5) SHELF LOCATION
6) POSSIBLE HAZARDS
7) QUANTITY ON HAND
8) ANNUAL NEEDS
9) CATALOG NUMBER
10) COST
TYPE IN THE NUMBER AFTER WHICH YOU WANT TO INSERT THE NEW
HEADING (OR TYPE 0 NOT TO ADD A NEW HEADING) ? 5
(user wants to insert before possible hazards)
WHAT DO YOU WANT TO CALL THE NEW HEADING ? BOILING POINT
CHOICES: 1) ADD A NEW HEADING
         2) CHANGE A HEADING NAME
            (AND/OR DATA UNDER HEADING)
         3) DELETE A HEADING
         4) LEAVE THIS PROGRAM
TYPE IN THE NUMBER OF YOUR CHOICE ? 4
(user is finished and exits the program)
```

If the user chooses option 3, the computer responds by printing: WHAT IS THE VALUE YOU WANT ALL DATA TO BE SET TO? and the user can specify the standard. If option 1 is chosen, the computer requests the value for each entry in the file in alphabetical order.

When changing a heading name, the options are similar:

1) CHANGE DATA FOR EACH ENTRY, 2) LEAVE DATA ALONE, 3) CHANGE ALL DATA TO THE SAME VALUE.

If option 1 is chosen, the value for each entry will be requested in alphabetical order. Option 2 leaves the data alone and only changes the heading name. Option 3 is the same as for adding a heading (all entries will be changed to a specified value under the heading which is being changed). (See note 2)

If the user decides to delete a heading, the computer will give you one last chance to change your mind before the deletion occurs so if you make a mistake in entering the heading number, it will not result in a disaster.

Notes On CHANG

1. Since adding a new heading increases the data file size, the user should be sure that the physical size of the file is large enough prior to running CHANG for this purpose. To increase the physical size either EXPAN or SPLIT can be used.

2. When deleting or changing the data under a heading, all the previously stored data is lost. The user should take care to use these features only when that data is no longer needed.

PROGRAMMER'S GUIDE

This section of the manual is designed to give those familiar with programming in BASIC a better idea of how the programs in the INFO system work. A discussion is provided on size of files and necessary memory to use the programs, the time it takes to run the programs, and a conversion guideline is presented to explain those BASIC commands which are not always available on other micro-computer systems. A section on each program presents the program listing, the function of each variable, and suggestions for modifications which might be desirable for other applications.

File Structure

The INFO programs were developed on a micro-computer which does not permit random access to records in the data file. All file input and output must be done sequentially which requires more time. Furthermore, the version of BASIC used permits only one read file to be open at any given time. This means the process of updating the data file takes longer since entries at the beginning of the file must be read each time the file is reopened.

The format of the data file is shown below. The variable HNUM refers to the number of headings or data categories used in the file.

RECORD NUMBER	DATA
1	number of headings (HNUM)
2	name of first heading
3	name of second heading
.	
.	
.	
HNUM+1	name of last heading
HNUM+2	first entry data for first heading
HNUM+3	first entry data for second heading
HNUM+4	first entry data for third heading
.	
.	
.	
2*HNUM+1	first entry data for last heading
2*HNUM+2	second entry data for first heading
.	
.	
.	
(N+1)*HNUM+1	Nth entry data for last heading
EOF mark	end-of-file

N is the number of entries currently in the file.

Our version of BASIC also restricts the number of characters which can be written or read in one operation to a maximum of 72, so no data item can exceed this Commas cannot be included in an input, since that character is reserved to separate data items.

The size of data files in our version of BASIC is determined in sectors. Each sector contains 256 bytes of usable storage space. The number of headings in the file is written and read as a number and occupies 5 bytes. Each heading name and each data item are alphanumeric strings and require the number of characters in that string plus 2 bytes to store. To calculate the approximate maximum number of sectors required to store a data file, use the formula:

```
((N+1)*HNUM*MAX+5)/256
```

where N is the number of entries to be stored in the file, HNUM is the number of headings or data categories for each entry, and MAX is the average number of characters stored under each data category. For example, if a file containing 100 entries, each with 10 data categories, is to be created and the average length for each category is 20 characters, the formula ((100+1)*10*20+5)/256 yields the approximate number of sectors (79) of storage that will be needed.

It may be desirable on some computers to change the file structure to permit random access to the data. If this is done, it may be necessary to store other data (such as the number of characters in the headings and the standard length for each data field) in the file. Once these numbers are known, it would be possible to address any item in the file without reading each previous item sequentially. This can decrease the time required to search the file for specific information, but may increase the amount of disk space required to store the data file.

Time Considerations

With sequential files and only one read file open at any given time, searching and updating the data file can take a great deal of time. For example, using the INFO program, it takes about one minute to search a data file consisting of 2000 data items (250 entries, each containing 8 data items). To make five changes which are not entered alphabetically, the program takes nearly 10 minutes with that size data file.

The SEARC and WORLD programs also can require much time, depending on the size of the data file and the search criteria. The WORLD program may require more than an hour to search a file such as the one described in the preceding paragraph.

These problems partially can be overcome by using the SPLIT program to separate a large data file into smaller ones which can be searched more rapidly. However, if it is possible to use random access files or to have more than one read file open, the programmer should give serious consideration to making the changes necessary to speed up program execution.

Memory Size

The longest program in the INFO system occupies slightly more than 24K (about 25,000) bytes of memory space. The amount of memory required for the variables used in each program varies according to the number of categories under which data is stored. In the present version of the INFO programs, the largest this requirement will be can be determined by the formula:

7700+520*HNUM

where HNUM is the number of headings or categories of data in the file. In the example where HNUM=10 the variables would require, at most, 7700+520*10=13200 bytes of memory.

Therefore, a total of 38,200 bytes of memory (a little over 37K) is needed, not counting any memory space required to load the BASIC program itself. If this requirement is too high for your micro-computer, the programs will have to be modified. Reducing the number of options or the maximum number of search criteria in SEARC, PSEAR, and WORLD will help. INFO can be split into several programs each with a specific purpose, (such as ADD, DELETE, etc.), and, if the CHAIN statement is available, these programs can be organized to run from a main "menu" program with a CHAIN to the routine for each single function and a CHAIN at the end of the routine back to the menu program.

Conversion (Transportability) Guidelines

Unfortunately, there is no one standardized version of the BASIC programming language. Because different versions of BASIC exist, it often is difficult to transport programs from one computer to another. Changes usually need to be made to the syntax of several programming statements before the programs will work. The programs in the INFO system were written on the IMSAI micro-computer using the DEBBI I version of BASIC.

The following commands may not be available in other versions of BASIC and can serve as a guide in identifying statements which might need revision to make the programs in the INFO system work on other computers. In the discussion N and N1 are assumed to be numeric values or variables, S$ and S1$ are strings or alphanumeric variables, and LN, LN1, and LN2 are assumed to represent line numbers within the program.

CLEAR N sets aside N words for handling strings during the course of program execution. Many BASIC'S do not require this statement and it can be left out. Some versions of BASIC require each string to be separately dimensioned which serves the same purpose in reserving string space.

WIDTH N informs BASIC that the output device (printer, terminal, etc.) may contain up to N characters per line. Some BASIC'S use a MARGIN statement or leave the width of the line up to the software to assure that lines do not run over.

OPENR S$:N opens file S$ for reading from the diskette in drive N. Many BASIC'S use statements such as FILE # N,S$ to open a file for reading on channel N. In these BASIC'S no distinction is made between files being used for writing (at the time the files are opened) and those used for reading. The INFO programs assume only one file to be open for reading and one for writing at any given time (a peculiarity of the version of BASIC used).

OPENW S$:N opens file S$ for writing on the diskette in drive N. See the discussion above.

CLOSR closes the file currently being read. (See the discussion below.)

CLOSW closes the file currently being written. Some BASIC'S have no statement equivalent to CLOSR or CLOSW. Other versions use a RESTORE # N statement.

DSKIN N (or S$) reads the next numeric (or string) record from the file currently open for reading. Some BASIC'S use statements such as READ # N,N1 or INPUT # N,S$. Note that when reading a record into a string variable (using either the INPUT or DSKIN statements) the INFO programs first set the string variable equal to the null string (""). This is done to ensure that a null string will be correctly read. In our version of BASIC, if a null record is read, the string variable is not altered from its previous value. In most BASIC'S there will be no need to first set the value of the string variable to the null string.

DSKOUT N (or S$) writes numeric (or string) records into the file currently open for writing. Some BASIC'S use statements like PRINT # N,N1 or WRITE # N,S$.

ON ERROR GOTO LN is used to test for end-of-file condition while attempting to read data from the input file currently open. If an error (end-of-file) occurs during an attempted read, control is transferred to the statement numbered LN. Some BASIC'S use statements such as IF EOF THEN LN. This problem can be overcome by having the software keep track of how many data items are in the input file and how many have been read so that the program

never requests a read operation unless there is still data left in the file.

IF ERR=27 THEN LN is used in conjunction with the ON ERROR statement to verify that the error which occurred really was an end-of-file. In our version of BASIC, the system variable ERR is set to a numeric value of 27 if an end-of-file error occurs. If no ON ERROR statement is available in your version of BASIC, this statement can be replaced with GOTO LN.

RESUME LN is used after an ON ERROR statement causes a jump in order to resume program execution at the next statement after the jump (LN). If no ON ERROR statement is used, this statement can be replaced with GOTO LN (or left out entirely providing you move the rest of the program up one line number).

MID$(S$,N,N1) is a function which returns the substring of string variable S$ beginning at position N and having length N1. If N1 is omitted, it returns the rest of the string starting at position N. This function often is called SUBSTR in other BASIC'S or is denoted differently (for example, S$(N,N1) is used in some versions of BASIC which do not permit string arrays).

ERASE eliminates arrays which are no longer needed or which must be redimensioned before further use. This command saves space, but some versions of BASIC do not offer this feature.

LPRINT is the same as a regular PRINT, but sends the data to the computer's printer port. If this statement is not available or your system does not have a printer, these commands may be replaced by PRINT and a paper-printing terminal can be used to obtain hard copy of program output.

TAB(N) causes the printing device to skip to the Nth column and resume printing at that position. This command can be replaced by a string of blanks if the format of the output need not be in column form. If column format is necessary, a loop to print spaces until the Nth column is reached can be substituted for the TAB command if it does not exist on your system.

IF N=1 THEN N1=1 ELSE N1=2 allows the execution of one or the other assignment statements for N1 depending on the value of N. If your system does not permit compound statements, replace such instructions by two conditionals such as IF N=1 THEN N1=1 and IF N >1 THEN N1=2. If your BASIC does not permit executable commands on conditionals, then use a line number after the THEN and place the assignment of N1 on that line.

LEN(S$) is a built-in function which returns the number of characters currently in S$.

VAL(S$) is a function which returns the numeric value of the numerals stored in S$. The string can contain only legitimate numeric characters or an error occurs.

STR$(N) is another function which returns the character representation of the numeric value stored in N. It can be considered the opposite of the VAL function.

LEFT$(S$,N) is a built-in function to return the first N characters of S$. Many BASIC versions do not support this function, but use expressions such as S$(1,N) to represent the substring.

RIGHT$(S$,N) returns the last N characters (or the rightmost) of the string S$. To simulate this function on other versions of BASIC, you can use S$(LEN(S$)+1-N,N).

RESTORE is used to set the pointer used in READing DATA statements back to the first value so the program can be run again. Some BASIC's may not have this feature which would make it difficult to have the program run more than once (without entering the RUN command again).

ON N GOTO LN,LN2,LN3 is used in routing. Depending on the value of the variable N, the computer branches to one of the line numbers LN, LN2, or LN3. For example, if N=2 then the computer would choose the second line number, LN2. If your system does not support such statements, use a series of conditionals to branch to the respective lines.

Our version of BASIC permits more than one statement to be written on one line if the two statements are separated by a colon. For example, X=3:Y=2 assigns values to each variable. If your version of BASIC does not permit compound statements, split these statements into two or more line numbers.

Note that the version of BASIC used in the INFO programs must permit string arrays and execution time dimensioning of arrays. If execution time dimensioning of arrays is not possible, set the array sizes to the maximum size possible. If string arrays are not permitted, the programmer will have to devise some method of making long strings usable in pieces (perhaps by using the substring function provided with the software). This may also necessitate a standard length for each data item in the file for easier addressing of the string.

Program Listings and Variables

The rest of this manual contains the program listings and other materials to help the programmer understand the logic used in the INFO system. In addition to the above conversion guidelines, the programmer should note that many BASIC systems do not permit variable names of 2 or more letters. In modifying these programs, it may be necessary to rename many of the variables. Most of the programs contain a routine toward the end to check the diskette in drive 0 for the existence of the EXEC program, which is our computer's disk operating system (sometimes called FDOS). This can be left out of your version or a similar check inserted for your own operating system.

SETUP

Possible modifications which would be necessary to permit random access would be to permit the user to specify the maximum field length for each data category or heading. These numbers (as well as the total number of entries in the current file and other such data) can be used to compute the byte location for any item in the file.

The following list contains the variables used in the SETUP program and the functions each variable serves.

Name	Description
FILE$	stores the name of the data file
HNUM	number of headings to be used in the file
I	loop parameter
NAME$	used to enter heading names

SETUP LISTING

```
10 REM *** PROGRAMMED BY VIK MUIZNIEKS 11/77
20 REM *** THIS PROGRAM CREATES A FILE TO BE USED WITH INFO PACKAGE
30 REM ***
40 PRINT "ALWAYS PRESS THE RETURN KEY AFTER TYPING-IN A RESPONSE!!!!!!"
50 PRINT "TYPE IN THE NAME OF THE FILE YOU WANT TO CREATE.  NOTE THAT"
60 PRINT "SPACE FOR THE FILE MUST HAVE BEEN ALLOCATED USING THE FDOS"
70 PRINT "ALLOC COMMAND, AND THE FILE MUST BE ON THE DISKETTE IN DRIV";
80 PRINT "E 1"
90 FILE$="": INPUT FILE$
100 IF FILE$="" THEN 50
110 ON ERROR GOTO 140
120 OPENW FILE$+":1"
130 GOTO 180
140 RESUME 150
150 PRINT "***ERROR***  TRY AGAIN."
160 CLOSE
170 GOTO 50
180 PRINT "HOW MANY HEADINGS WILL BE IN YOUR FILE";
190 INPUT HNUM
200 IF HNUM < 1 THEN 180
210 DSKOUT HNUM
220 FOR I=1 TO HNUM
230 PRINT "WHAT DO YOU WANT TO CALL HEADING NUMBER ";I
240 NAME$="": INPUT NAME$
250 IF NAME$ <> "" THEN 280
260 PRINT "YOU MUST TYPE IN SOMETHING!"
270 GOTO 230
280 DSKOUT NAME$
290 NEXT I
300 CLOSW
310 PRINT "FILE ";FILE$;" IS READY TO BE USED WITH OTHER PROGRAMS."
320 END
```

INFO

The search algorithm (line numbers 970-1270, et al.) would need modification for random access in order to speed up the process of locating a specific entry. In the present version, since only one read file may be open, the data file is left open after a search is performed to save time and the next search starts at that point unless the entry which is sought has a smaller value than the next entry to be read. A binary search could be used if random access is permitted. Another option could be added to this program to permit changes to be entered if mistakes are noticed while viewing an entry.

The process of updating the data file involves alphabetizing the changes which were entered (line numbers 4610-4870), copying the file while making the changes (4880-6210) and copying the updated file back into the permanent data file (6220-6360). This requires more time than is needed if either random access or the ability to have more than one read file open at any time are available in your version of BASIC.

If memory space is limited, this program could be split into several smaller programs each permitting only one option (such as add or delete).

Searches for equal, less than, or greater than entries are done by comparing ASCII character codes for the items being searched. The chart below shows the values of some of the characters:

ASCII Codes

CODE	CHARACTER
0	NUL
10	LF (line feed)
12	FF (page or form feed)
13	CR (carriage return)
32	SPACE (blank)
33	! (exclamation point)
34	" (quotation mark)
36	$ (dollar sign)
37	% (per cent sign)
38	& (ampersand)
39	' (apostrophe)
40	((left paren)
41	) (right paren)
45	- (hyphen)
46	. (period)
47	/ (slash)
48-57	digits 0-9
58	: (colon)

59	; (semi-colon)
63	? (question mark)
65-90	capital letters A-Z
97-122	lower case letters, a-z

The following list contains the variables used in the INFO program and the function each variable serves.

Name	Description
A$	used to input answers and disk info
CHOICE	used to input user choices for routing through the program (sometimes abbreviated as CH)
COUNT	counts number of lines displayed on the terminal
EF	flag set to 1 if update is needed
FC$	holds the function of next entry not yet written while updating
FILE$	stores name of data file
FLAG	flag used to determine end of file (0 value); also used by subroutines to indicate whether a specific entry was located (0 value)
FUNC$	used to output the desired function for updating an entry
HDNAME$(HNUM)	used to read in heading names
HNUM	number of headings in the data file
I	loop parameter
I1	used to input the header number to be changed
L	represents the length of NAME$
LNAME$	used to store the last name read from the data file; later used to store the last name written in updating
M$	system message
NAME$	used to input the name of an entry
PAGESIZE	number of lines which can be displayed on the terminal
SCR$	stores the name of SCR1 file (on drive 0)
TEMP$(HNUM)	stores data on the entry currently being processed
UP$	the name of SCR file (on drive 0)

INFO LISTING

```
10 CLEAR 1000  'RESERVES STRING SPACE
20 PAGESIZE=14  'NUMBER OF USABLE LINES ON TERMINAL
30 EF=0  'FLAG SET TO 1 IF UPDATE NEEDED
40 SCR$="SCR1.0"  'SCRATCH FILE NAME
50 UP$="SCR.0"  'ALPHABETICAL SCRATCH FILE NAME FOR UPDATE
60 M$="PRESS THE RETURN KEY FOR MORE  TYPE ANYTHING ELSE TO STOP."  'SYS
70 GOSUB 170  'GET DATA FILE NAME
80 GOSUB 250  'OPEN SCRATCH FILE FOR OUTPUT
90 GOSUB 350  'OPEN DATA FILE AND INPUT HEADINGS
100 PRINT
105 REM MAIN BRANCHING POINT
110 PRINT"CHOICES: 1) LOOK AT DATA FILE, 2) ADD, 3) ALTER,"
120 PRINT"          4) DELETE, 5) STOP THE PROGRAM."
130 PRINT"TYPE THE NUMBER OF THE CHOICE YOU WANT."
140 INPUT CHOICE  'GETS USER OPTION
150 IF (CH-1)*(CH-2)*(CH-3)*(CH-4)*(CH-5)<>0 THEN 100  'CHECK VALIDITY
160 ON CHOICE GOTO 510,1580,2380,3560,3790  'BRANCH TO OPTION
165 REM SUBROUTINE TO INPUT DATA FILE NAME
170 PRINT"FILES SCR AND SCR1 SHOULD BE ON THE DISKETTE IN DRIVE 0."
180 PRINT"IF THEY ARE NOT, CHANGE THE DISKETTE NOW!!!"
190 PRINT"WHAT IS THE NAME OF THE DATA FILE? (SHOULD BE IN DRIVE 1."
200 FILE$=""  'NAME OF DATA FILE
210 INPUT FILE$
220 IF FILE$="" THEN 190  'NO NAME INPUT, TRY AGAIN
230 FILE$=FILE$+",1"  'DATA FILE MUST BE IN DRIVE 1
240 RETURN
245 REM SUBROUTINE TO OPEN SCR1 TO WRITE CHANGES FOR DATA FILE UPDATE
250 ON ERROR GOTO 280  'NO SCRATCH FILE IN DRIVE 0
260 OPEN SCR$  'OPEN FILE
270 GOTO 340  'NORMAL SUCCESSFUL END OF ROUTINE
280 RESUME 290  'FAILURE ROUTINE
290 CLOSE  'REINITIALIZE AND TRY AGAIN
300 PRINT"PLACE THE DISKETTE WITH FILES SCR AND SCR1 IN DRIVE 0."
310 PRINT"THEN PRESS THE RETURN KEY."
320 INPUT A$  'INPUT RETURN KEY BEFORE CONTINUING
330 GOTO 240  'START AGAIN
340 RETURN
345 REM SUBROUTINE TO OPEN DATA FILE AND INPUT HEADINGS
350 ON ERROR GOTO 450  'FILE NOT FOUND OR WRONG FORMAT
360 OPENR FILE$  'OPEN DATA FILE TO READ
370 DSKIN HNUM  'NUMBER OF HEADINGS IN DATA FILE
380 DIM HNAME$(HNUM)  'NAMES OF HEADINGS
390 DIM TEMP$(HNUM)  'DATA STORAGE
400 TEMP$(1)=""
410 FOR I=1 TO HNUM  'INPUT HEADINGS
420 DSKIN HNAME$(I)
430 NEXT
440 GOTO 500  'NORMAL SUCCESSFUL END OF ROUTINE
450 RESUME 460  'DATA FILE NOT FOUND
460 PRINT"PLACE THE DISKETTE WITH THE DATA FILE IN DRIVE 1."
470 PRINT"THEN PRESS THE RETURN KEY."
480 INPUT A$  'INPUT RETURN BEFORE CONTINUING
490 GOTO 360  'START AGAIN
500 RETURN
505 REM LOOK AT DATA FILE ROUTINE
510 PRINT
515 REM BRANCHING POINT
520 PRINT"CHOICES: 1) LIST OF NAMES, 2) DATA ON ONE ENTRY,"
530 PRINT"         3) DATA ON ALL ENTRIES, 4) NONE OF THESE."
540 PRINT"TYPE THE NUMBER OF YOUR CHOICE."
550 INPUT CHOICE  'INPUTS USER OPTION
560 IF (CH-1)*(CH-2)*(CH-3)*(CH-4)<>0 THEN 510  'CHECK VALIDITY
570 ON CHOICE GOTO 580,940,1300,100  'BRANCH TO OPTION
580 IF TEMP$(1)<>"" THEN GOSUB 730  'CHECK CURRENT FILE POSITION
585 REM ROUTINE TO OUTPUT NAMES IN DATA FILE
590 PRINT
600 FLAG=1  'SET TO 0 WHEN END-OF-FILE ENCOUNTERED
610 COUNT=0  'NUMBER OF LINES OUTPUT SO FAR
620 GOSUB 820  'INPUT ONE RECORD
630 IF FLAG=0 THEN 510  'CHECK FOR EOF--CONTINUE IF FILE NOT DONE
640 IF COUNT<PAGESIZE THEN 700  'CHECK TERMINAL SIZE BEFORE OUTPUTTING
650 PRINT M$  'PRINT MESSAGE IF SCREEN FULL
660 A$=""
670 INPUT A$  'INPUT RETURN BEFORE CONTINUING
680 IF A$<>"" THEN 510  'IF USER INPUTS ANYTHING ELSE START AGAIN
690 COUNT=0  'START NEW PAGE
700 COUNT=COUNT+1
710 PRINT TEMP$(1)  'OUTPUT NEXT NAME
720 GOTO 620  'INPUT NEXT RECORD
725 REM SUBROUTINE TO REINITIALIZE DATA FILE
730 ON ERROR GOTO 750  'UNKNOWN PROBLEM
740 CLOSE
750 RESUME 760  'CONTINUE
760 OPENR FILE$  'OPEN DATA FILE AGAIN
770 DSKIN HNUM  'READ IN NUMBER OF HEADINGS AGAIN
780 FOR I=1 TO HNUM  'SKIP OVER HEADING NAMES
790 DSKIN A$
800 NEXT
810 RETURN
815 REM SUBROUTINE TO READ ONE RECORD OF DATA
820 ON ERROR GOTO 900  'END OF FILE ERROR
830 TEMP$(1)=""
840 DSKIN TEMP$(1)  'GET NEXT NAME
850 FOR I=2 TO HNUM  'INPUT REST OF RECORD
860 TEMP$(I)=""
870 DSKIN TEMP$(I)
```

INFO CONTINUED

```
880 NEXT
890 GOTO 930   'SUCCESSFUL ENDING OF ROUTINE
900 RESUME 910   'FAILURE DUE TO EOF
910 GOSUB 730   'REINITIALIZE DATA FILE
920 FLAG=0   'SET FLAG TO INDICATE EOF
930 RETURN
935 REM ROUTINE TO FIND ONE RECORD
940 PRINT HDNAME$(1);"? (PRESS RETURN FOR NONE )"   'GET RECORD NAME
950 NAME$=""
960 INPUT NAME$   'STORES NAME OF RECORD
970 IF NAME$="" THEN 510   'IF USER PRESSED RETURN THEN RETURN TO BRANCH
980 PRINT
990 IF TEMP$(1)<NAME$ THEN 1010   'CHECK DATA FILE LOCATION
995 REM CONTINUE IF NOT PAST NAME--OTHERWISE REINITIALIZE DATA FILE
1000 GOSUB 730   'REINITIALIZE DATA FILE
1010 FLAG=1   'SET TO 0 IF RECORD IS FOUND
1020 GOSUB 1200   'TRY TO FIND THE RECORD
1030 IF FLAG=0 THEN 1060   'CHECK TO SEE IF FOUND
1040 PRINT NAME$;" WAS NOT FOUND "
1050 GOTO 510   'RETURN FOR ANOTHER OPTION IF NOT FOUND
1060 COUNT=0   'PRINT RECORD IF FOUND
1070 FOR I=1 TO HNUM   'PRINT EACH FIELD OF DATA
1080 PRINT HDNAME$(I);"=";TEMP$(I)
1090 COUNT=COUNT+1
1100 IF COUNT<PAGESIZE THEN 1180   'START NEW PAGE IF NECESSARY
1110 PRINT M$   'OUTPUT MESSAGE IF PAGE FULL
1120 A$=""
1130 INPUT A$
1140 IF A$<>"" THEN 1170   'ESCAPE IF USER PRESSES ANYTHING EXCEPT RETURN
1150 COUNT=0   'START NEW PAGE
1160 GOTO 1180   'CONTINUE PRINTING
1170 I=HNUM   'ESCAPE IF USER PRESSED ANYTHING EXCEPT RETURN
1180 NEXT
1190 GOTO 510   'RETURN FOR ANOTHER USER OPTION
1195 REM SUBROUTINE TO LOCATE RECORD
1200 L=LEN(NAME$)   'FIND LENGTH OF DESIRED NAME
1210 GOSUB 820   'READ NEXT RECORD
1220 IF FLAG=0 THEN 1280   'FAILS IF EOF ENCOUNTERED
1230 IF LEFT$(TEMP$(1),L)=NAME$ THEN 1260   'FOUND IT
1240 IF TEMP$(1)>NAME$ THEN 1280   'WENT PAST IT
1250 GOTO 1210
1260 FLAG=0   'SET FLAG FOR SUCCESSFUL READ
1270 GOTO 1290
1280 FLAG=1   'SET FLAG FOR UNSUCCESSFUL READ
1290 RETURN
1295 REM ROUTINE TO PRINT ALL DATA IN FILE
1300 FLAG=1   'SET TO 0 ON EOF
1310 IF TEMP$(1)="" THEN 1350   'REINITIALIZE DATA FILE IF NOT AT START
1320 GOSUB 730   'REINITIALIZE DATA FILE
1330 A$="CONT"   'USED TO CONTINUE PRINTING DATA
1340 COUNT=0
1350 GOSUB 820   'READ NEXT RECORD
1360 IF FLAG=0 THEN 510   'ON EOF RETURN FOR NEXT OPTION
1370 PRINT
1380 FOR I=1 TO HNUM   'OUTPUT NEXT RECORD
1390 IF I<>1 THEN 1420   'SKIP EXTRA LINE EXCEPT BETWEEN RECORDS
1400 COUNT=COUNT+1
1410 PRINT
1420 PRINT HDNAME$(I);"=";TEMP$(I)   'OUTPUT NEXT LINE OF DATA
1430 COUNT=COUNT+1
1440 IF COUNT<PAGESIZE THEN 1540   'IS PAGE FULL?
1450 PRINT M$   'OUTPUT MESSAGE IF SO
1460 A$=""
1470 INPUT A$   'INPUT RETURN BEFORE CONTINUING
1480 IF A$<>"" THEN 1530   'END IF USER INPUTS ANYTHING EXCEPT RETURN
1490 A$="CONT"
1500 COUNT=0   'START NEW PAGE
1510 IF I=1 THEN 1420   'PRINT NAME AGAIN IF AT START OF RECORD
1520 GOTO 1540
1530 I=HNUM   'TERMINATE LOOP IF DATA NOT WANTED
1540 NEXT
1550 IF A$="CONT" THEN 1350   'CONTINUE PRINTING
1560 IF A$<>"" THEN 510   'RETURN FOR ANOTHER OPTION
1570 GOTO 1350   'CONTINUE PRINTING
1575 REM ROUTINE TO ADD NEW RECORD
1580 FUNC$="ADD"   'INDICATES FUNCTION FOR EACH RECORD
1590 PRINT HDNAME$(1);"? (PRESS RETURN FOR NONE )"   'GET NEW NAME
1600 NAME$=""
1610 INPUT NAME$   'STORES NEW NAME
1620 IF NAME$="" THEN 100   'RETURN TO MAIN BRANCHING POINT IF USER PRESS
1630 IF TEMP$(1)<NAME$ THEN 1650   'CHECK DATA FILE POSITION
1640 GOSUB 730   'REINITIALIZE IF PAST NEW NAME
1650 FLAG=1   'SET TO 0 IF RECORD ALREADY EXISTS UNDER NEW NAME
1660 GOSUB 1200   'FIND NAME IF IT EXISTS
1670 IF FLAG=1 THEN 1700   'SEE IF NAME IS REALLY NEW
1680 PRINT NAME$;" ALREADY EXISTS  TRY AGAIN "   'NAME IS OLD
1690 GOTO 1590   'TRY AGAIN
1700 L=NAME$=TEMP$(1)   'STORE CURRENT DATA FILE POSITION
1710 TEMP$(1)=NAME$   'STORE NEW NAME
1720 FF=1   'SET FLAG TO INDICATE UPDATE NEEDED
1730 PRINT"ENTER YOUR DATA "
1740 FOR I=2 TO HNUM   'INPUT REST OF RECORD
1750 PRINT HDNAME$(I);"=";
1760 TEMP$(I)=""
1770 INPUT TEMP$(I)
1780 NEXT
1790 PRINT"THIS IS YOUR LAST CHANCE TO CHECK THE DATA YOU JUST INPUT "
1800 FOR I=1 TO HNUM   'PRINT RECORD FOR CHANGES
```

INFO CONTINUED

```
1810 PRINT I;HDNAME$(I);"=";TEMP$(I)
1820 NEXT
1830 PRINT"DO YOU WANT TO CHANGE ANY OF THESE ITEMS? (YES OR NO)"
1840 NAME$=""
1850 INPUT NAME$  'INPUT RESPONSE FOR CHANGES
1860 IF LEFT$(NAME$,1)="N" THEN 1980
1870 IF LEFT$(NAME$,1)<>"Y" THEN 1830
1880 PRINT"TYPE THE NUMBER OF THE ITEM YOU WANT TO CHANGE."
1885 REM ROUTINE TO PERMIT CHANGES TO NEW RECORD
1890 INPUT CH  'INPUT NUMBER OF DATA ITEM TO BE CHANGED
1900 IF CH<1 THEN 1880  'CHECK VALIDITY
1910 IF CH>HNUM THEN 1880
1920 IF CH<>INT(CH) THEN 1880
1930 IF CH=1 THEN 2040  'TAKE CARE OF NAME CHANGE
1935 REM ROUTINE TO INPUT OTHER CHANGES
1940 PRINT"ENTER THE CORRECTED INFORMATION FOR ";HDNAME$(CH)
1950 TEMP$(CH)=""
1960 INPUT TEMP$(CH)
1970 GOTO 1790  'OUTPUT RECORD AND SEE IF OTHER CHANGES NEEDED
1975 REM ROUTINE TO DUMP RECORD TO SCRATCH FILE WHEN DONE
1980 DSKOUT FUNC$  'STORE FUNCTION
1990 FOR I=1 TO HNUM  'STORE DATA RECORD
2000 DSKOUT TEMP$(I)
2010 NEXT
2020 TEMP$(1)=LNAME$  'RESTORE POINTER TO DATA FILE POSITION
2030 GOTO 1590  'GET NEXT NEW NAME
2035 REM ROUTINE TO HANDLE NAME CHANGE IN NEW RECORD
2040 PRINT"ENTER THE CORRECTED INFORMATION FOR ";HDNAME$(1)
2050 NAME$=""
2060 INPUT NAME$  'GET NEW NAME
2070 IF NAME$="" THEN 1790  'USER INPUT RETURN SO OUTPUT RECORD AGAIN
2080 FLAG=1  'FLAG SET TO 0 IF NAME ALREADY EXISTS IN DATA FILE
2090 GOSUB 2150  'SEE IF NEW NAME EXISTS
2100 IF FLAG=1 THEN 2130  'CHECK RESULTS OF SEARCH FOR NEW NAME
2110 PRINT NAME$;" ALREADY EXISTS.  TRY AGAIN."
2120 GOTO 1790  'NAME ALREADY EXISTS SO OUTPUT RECORD AND TRY AGAIN
2130 TEMP$(1)=NAME$  'NAME DOES NOT EXIST SO STORE IT
2140 GOTO 1790  'OUTPUT RECORD FOR FURTHER CHANGES
2145 REM SUBROUTINE TO CHECK EXISTENCE OF NEW NAME
2150 L=LEN(NAME$)  'FIND LENGTH OF NAME
2160 IF LNAME$<NAME$ THEN 2180  'REINITIALIZE FILE IF ALREADY PAST NEW N
2170 GOSUB 730  'REINITIALIZE DATA FILE IF NECESSARY
2180 GOSUB 2270  'READ ONE RECORD
2190 IF FLAG=0 THEN 2250  'CHECK FOR EOF
2200 IF LEFT$(LNAME$,L)=NAME$ THEN 2230  'CHECK FOR EXISTENCE OF NAME
2210 IF LNAME$>NAME$ THEN 2250  'ALREADY PAST IT
2220 GOTO 2180  'GET NEXT RECORD
2230 FLAG=0  'SET FLAG FOR RECORD FOUND
2240 GOTO 2260  'ABNORMAL RETURN FOR RECORD FOUND
2250 FLAG=1  'SET FLAG FOR RECORD NOT FOUND
2260 RETURN
2265 REM SUBROUTINE TO READ ONE RECORD WHEN CHECKING FOR NAME EXISTENCE
2270 ON ERROR GOTO 2340  'END OF FILE
2280 LNAME$=""
2290 DSKIN LNAME$  'READ IN NEXT RECORD NAME
2300 FOR I=2 TO HNUM  'SKIP REST OF RECORD
2310 DSKIN A$
2320 NEXT
2330 GOTO 2370  'NORMAL END OF SUBROUTINE
2340 RESUME 2350  'END OF FILE
2350 GOSUB 730  'REINITIALIZE DATA FILE
2360 FLAG=0  'SET FLAG TO INDICATE EOF
2370 RETURN
2375 REM ROUTINE TO CHANGE AN OLD RECORD IN DATA FILE
2380 FUNC$="ALT"  'SET FUNCTION NAME TO ALTER
2390 PRINT"WHICH ";HDNAME$(1);" DO YOU WANT TO CHANGE?"
2400 PRINT"PRESS RETURN FOR NONE "
2410 NAME$=""
2420 INPUT NAME$  'GET NAME OF RECORD TO BE CHANGED
2430 IF NAME$="" THEN 100  'USER INPUT BLANK NAME SO RETURN FOR NEXT OPT
2440 IF TEMP$(1)<NAME$ THEN 2460  'CHECK DATA FILE POSITION
2450 GOSUB 730  'REINITIALIZE IF PAST NAME
2460 FLAG=1  'SET TO 0 IF RECORD FOUND
2470 GOSUB 1200  'TRY TO FIND RECORD
2480 IF FLAG=0 THEN 2510  'CHECK TO SEE IF RECORD FOUND
2490 PRINT NAME$;" WAS NOT FOUND.  TRY AGAIN."
2500 GOTO 2390  'RECORD NOT FOUND SO GET ANOTHER NAME
2510 PRINT TEMP$(1)  'NOTIFY USER THAT NAME WAS FOUND
2520 LNAME$=TEMP$(1)  'STORE CURRENT POSITION IN DATA FILE
2530 PRINT
2535 REM BRANCHING POINT
2540 PRINT"CHOICES:  1) CHANGE ONE HEADING.  2) CHANGE MORE THAN ONE."
2550 PRINT"          HEADING.  3) SEE EACH HEADING AND CHANGE ANY"
2560 PRINT"          THAT YOU WANT."
2570 PRINT"TYPE THE NUMBER OF THE CHOICE YOU WANT."
2580 INPUT CH  'GET USER OPTION
2590 IF (CH-1)*(CH-2)*(CH-3)<>0 THEN 2510  'CHECK VALIDITY
2600 ON CHOICE GOTO 2610,2610,3200  'BRANCH TO CHOICE
2610 PRINT"WHICH HEADING NUMBER DO YOU WANT TO CHANGE?"
2620 PRINT"TYPE 0 FOR A LIST OF HEADINGS."
2630 INPUT I1  'GET NUMBER OF HEADING TO BE CHANGED
2640 IF I1<>0 THEN 2670
2650 GOSUB 3050  'PRINT HEADINGS IF USER ENTERS 0
2660 GOTO 2610  'GO BACK TO GET HEADING NUMBER
2670 IF I1<>1 THEN 2910  'CHECK TO SEE IF NAME IS TO BE CHANGED
2675 REM ROUTINE TO HANDLE NAME CHANGE
2680 PRINT"ENTER THE CORRECTED INFORMATION FOR ";HDNAME$(1)
2690 NAME$=""
```

INFO CONTINUED

```
2700 INPUT NAME$   'GET NEW NAME
2710 IF NAME$="" THEN 2610   'USER PRESSED ONLY RETURN SO TRY AGAIN
2720 FLAG=1   'SET TO 0 IF NAME ALREADY EXIST
2730 GOSUB 2150   'SEARCH FOR NEW NAME
2740 IF FLAG=1 THEN 2770   'CHECK TO SEE IF RECORD FOUND
2750 PRINT NAME$;" ALREADY EXISTS  TRY AGAIN "
2760 GOTO 2610   'CANNOT CHANGE TO NAME WHICH ALREADY EXISTS SO TRY AGAIN
2770 FUNC$="DEL"   'CHANGE FUNCTION TO DELETE OLD RECORD
2780 DSKOUT FUNC$   'STORE FUNCTION
2790 DSKOUT TEMP$(1)   'STORE OLD NAME
2800 TEMP$(1)=NAME$   'STORE NEW NAME
2810 FUNC$="ADD"   'SET FUNCTION TO ADD NEW RECORD FOR NEW NAME
2820 FF=1   'SET FLAG TO INDICATE UPDATE OF DATA FILE NEEDED
2830 IF CH=2 THEN 2990   'CONTINUE CHANGING IF USER CHOSE OPTION 2
2835 REM ROUTINE TO STORE CHANGED RECORD WHEN DONE
2840 DSKOUT FUNC$   'STORE FUNCTION
2850 FOR I=1 TO HNUM   'STORE RECORD
2860 DSKOUT TEMP$(I)
2870 NEXT
2880 TEMP$(1)=LNAME$   'RESTORE POINTER TO DATA FILE POSITION
2890 FF=1   'SET FLAG TO INDICATE DATA FILE UPDATE NEEDED
2900 GOTO 2380   'START AGAIN
2910 IF I1>HNUM THEN 2610   'CHECK VALIDITY OF HEADING NUMBER
2920 IF I1<0 THEN 2610
2930 IF I1<>INT(I1) THEN 2610
2940 PRINT HDNAME$(I1);"=";TEMP$(I1)
2950 PRINT"ENTER THE CORRECTED INFORMATION FOR ";HDNAME$(I1)
2960 TEMP$(I1)=""
2970 INPUT TEMP$(I1)   'INPUT CHANGES TO DATA ALREADY IN FILE
2980 IF CH=1 THEN 2840   'DUMP RECORD IF ONLY ONE CHANGE TO BE MADE
2990 PRINT"DO YOU WANT TO CHANGE ANOTHER ITEM? (YES OR NO)"
3000 A$=""
3010 INPUT A$   'INPUT RESPONSE TO SEE IF MORE CHANGES ARE NEEDED
3020 IF LEFT$(A$,1)="Y" THEN 2610
3030 IF LEFT$(A$,1)="N" THEN 2840
3040 GOTO 2990
3045 REM SUBROUTINE TO PRINT HEADING NAMES AND NUMBERS
3050 COUNT=0
3060 PRINT"HEADING NUMBERS AND NAMES:"
3070 FOR I=1 TO HNUM   'PRINT HEADING NAMES AND NUMBERS
3080 PRINT I;HDNAME$(I)
3090 COUNT=COUNT+1
3100 IF COUNT<PAGESIZE THEN 3180   'SEE IF PAGE FULL
3110 PRINT M$   'PRINT MESSAGE IF PAGE FULL
3120 A$=""
3130 INPUT A$   'INPUT RETURN BEFORE CONTINUING
3140 IF A$<>"" THEN 3170   'CONTINUE UNLESS USER HIT SOMETHING OTHER THAN
3150 COUNT=0   'START NEW PAGE
3160 GOTO 3180   'CONTINUE PRINTING
3170 I=HNUM   'ESCAPE IF USER HIT SOMETHING OTHER THAN RETURN
3180 NEXT
3190 RETURN
3195 REM ROUTINE TO OUTPUT EACH DATA ITEM  AND ALLOW CHANGES TO ANY
3200 PRINT"EACH ITEM WILL BE DISPLAYED.  IF YOU WANT TO CHANGE THAT"
3210 PRINT"ITEM, PRESS THE LETTER C AFTER THE QUESTION MARK.  IF YOU"
3220 PRINT"WANT TO LEAVE THE ITEM AS IT APPEARS, PRESS ONLY RETURN."
3230 PRINT HDNAME$(1);"=";TEMP$(1)   'PRINT NAME OF RECORD
3240 PRINT"TYPE C TO CHANGE OR PRESS THE RETURN KEY."
3250 A$=""
3260 INPUT A$   'GET OPTION
3270 IF A$<>"C" THEN 3490   'CHECK IF CHANGE IS NEEDED
3280 PRINT"TYPE IN THE CORRECTED INFORMATION"
3290 NAME$=""
3300 INPUT NAME$   'GET NEW NAME IF USER WANTS TO CHANGE IT
3310 IF NAME$="" THEN 3490   'BLANK NAME SO CONTINUE OUTPUT OF REST OF RE
3320 FLAG=1   'SET TO 0 IF NEW NAME IS FOUND
3330 GOSUB 2150   'SEARCH FOR NEW NAME
3340 IF FLAG=1 THEN 3370   'CHECK IF NAME WAS FOUND
3350 PRINT NAME$;" ALREADY EXISTS.  TRY AGAIN "
3360 GOTO 3230   'NEW NAME ALREADY EXISTS SO TRY AGAIN
3370 FUNC$="DEL"   'NEW NAME DOES NOT EXIST SO CHANGE FUNCTION
3380 DSKOUT FUNC$   'STORE FUNCTION TO DELETE OLD RECORD
3390 DSKOUT TEMP$(1)   'STORE OLD NAME
3400 TEMP$(1)=NAME$   'STORE NEW NAME
3410 FF=1   'SET FLAG TO INDICATE DATA FILE UPDATE NEEDED
3420 FUNC$="ADD"   'SET FUNCTION TO ADD NEW RECORD
3430 FOR I=2 TO HNUM   'CHECK FOR CHANGES OF OTHER ITEMS IN RECORD
3440 PRINT HDNAME$(I);"=";TEMP$(I)
3450 PRINT"TYPE C TO CHANGE OR PRESS THE RETURN KEY."
3460 A$=""
3470 INPUT A$   'INPUT RESPONSE FOR CHANGES
3480 IF A$<>"C" THEN 3530   'CHECK TO SEE IF ITEM IS TO BE CHANGED
3490 PRINT"TYPE IN THE CORRECTED INFORMATION."
3500 TEMP$(I)=""
3510 INPUT TEMP$(I)   'INPUT CHANGED DATA ITEM
3520 GOTO 3540   'CONTINUE WITH NEXT ITEM
3530 IF A$<>"" THEN 3450   'CHECK VALIDITY OF RESPONSE
3540 NEXT
3550 GOTO 2840   'DUMP RECORD WHEN FINISHED
3555 REM ROUTINE TO DELETE OLD RECORD IN DATA FILE
3560 FUNC$="DEL"   'SET FUNCTION TO DELETE
3570 PRINT"WHICH ENTRY DO YOU WANT TO DELETE? (PRESS RETURN FOR NONE)"
3580 NAME$=""
3590 INPUT NAME$   'GET NAME OF RECORD TO BE DELETED
3600 IF NAME$="" THEN 100   'BLANK NAME SO GET NEXT OPTION
3610 IF TEMP$(1)<NAME$ THEN 3630   'CHECK POSITION IN DATA FILE
3620 GOSUB 730   'REINITIALIZE IF PAST THE NAME
3630 FLAG=1   'SET TO 0 IF RECORD FOUND
3640 GOSUB 1200   'SEARCH FOR RECORD
3650 IF FLAG=0 THEN 3680   'SEE IF RECORD FOUND
```

INFO CONTINUED

```
3660 PRINT NAME$;" WAS NOT FOUND "
3670 GOTO 3570  'NAME WAS NOT FOUND IN DATA FILE SO TRY AGAIN
3680 PRINT"THIS WILL DESTROY ALL DATA ON ";TEMP$(1)
3690 PRINT"ARE YOU SURE YOU WANT TO DELETE THIS ENTRY? (YES OR NO)"
3700 A$=""
3710 INPUT A$  'INPUT LAST CHANCE TO CANCEL DELETION
3720 IF LEFT$(A$,1)="Y" THEN 3750
3730 IF LEFT$(A$,1)="N" THEN 3570
3740 GOTO 3680
3750 DSKOUT FUNC$  'DELETION STORED
3760 DSKOUT TEMP$(1)  'NAME STORED TO BE DELETED
3770 EF=1  'SET FLAG TO INDICATE DATA FILE UPDATE NEEDED
3780 GOTO 3570  'GET NEXT NAME
3785 REM ROUTINE TO UPDATE DATA FILE IF NEEDED
3790 IF EF=0 THEN 5580  'SEE IF UPDATE NEEDED
3800 PRINT"THE DATA FILE WILL NOW BE UPDATED  THIS MAY TAKE"
3810 PRINT"A LONG TIME  LEAVE THE MACHINE RUNNING UNTIL YOU"
3820 PRINT"SEE THE FOLLOWING MESSAGE PRINTED:"
3830 PRINT
3840 PRINT"INFO IS NOW FINISHED!!"
3850 PRINT"OK"
3860 PRINT
3870 PRINT"IF YOU TURN OFF THE COMPUTER BEFORE THIS MESSAGE,"
3880 PRINT"YOU RISK DESTROYING YOUR DATA FILE!"
3890 CLOSW  'CLOSE SCRATCH FILE
3900 CLOSR  'CLOSE DATA FILE
3910 OPENR SCR$  'OPEN SCRATCH FILE TO CHECK FOR ALPHABETICAL ORDER
3920 FLAG=0  'SET TO 1 IF FILE NOT ALPHABETICAL
3930 ON ERROR GOTO 4110  'END OF FILE ERROR
3940 FUNC$=""
3950 DSKIN FUNC$  'INPUT FUNCTION
3960 NAME$=""
3970 DSKIN NAME$  'INPUT NAME
3980 IF FUNC$="DEL" THEN 4030  'CHECK FUNCTION TO SEE IF RECORD NEEDS TO
3990 FOR I=2 TO HNUM  'INPUT REST OF RECORD
4000 A$=""
4010 DSKIN A$
4020 NEXT
4030 FUNC$=""
4040 A$=""
4050 DSKIN FUNC$  'GET NEXT FUNCTION
4060 DSKIN A$  'GET NEXT NAME
4070 IF A$<NAME$ THEN 4100  'CHECK ALPHABETICAL ORDER
4080 NAME$=A$  'CONTINUE IF IN ORDER
4090 GOTO 3980
4100 FLAG=1  'SET FLAG IF OUT OF ORDER
4110 RESUME 4120  'END OF FILE
4120 IF FLAG=0 THEN 4570  'SEE IF FILE IS ALPHABETIZED
4130 CLOSR  'NOT IN ORDER SO CLOSE SCRATCH FILE
```



```
4140 OPENR SCR$  'OPEN SCRATCH FILE TO ALPHABETIZE
4150 OPENW UP$  'OPEN UPDATE SCRATCH FILE TO STORE ALPHABETICAL CHANGES
4160 LNAME$="""  'STORES LAST NAME TO BE WRITTEN INTO UPDATE FILE
4170 ON ERROR GOTO 4430  'END OF FILE ERROR
4180 FC$=""
4190 DSKIN FC$  'READ FUNCTION
4200 TEMP$(1)=""
4210 DSKIN TEMP$(1)  'READ NAME
4220 IF FC$="DEL" THEN 4280  'SKIP TO NEXT RECORD IF DELETE
4230 FOR I=2 TO HNUM  'IF NOT DELETE READ REST OF RECORD
4240 FOR I=2 TO HNUM  'READ REST OF RECORD
4250 TEMP$(I)=""
4260 DSKIN TEMP$(I)
4270 NEXT
4280 FUNC$=""
4290 DSKIN FUNC$  'GET NEXT FUNCTION
4300 A$=""
4310 DSKIN A$  'READ NEXT NAME
4320 IF A$>=TEMP$(1) THEN 4370  'FIND SMALLEST RECORD NOT WRITTEN YET
4330 IF A$<=LNAME$ THEN 4370
4340 TEMP$(1)=A$  'WHEN FOUND STORE IN TEMP
4350 FC$=FUNC$  'STORE FUNCTION ALSO
4360 GOTO 4220  'CONTINUE THROUGH ENTIRE FILE
4370 IF FUNC$="DEL" THEN 4280  'SKIP TO NEXT RECORD IF DELETE
4380 FOR I=2 TO HNUM  'SKIP REST OF RECORD IF NOT DELETE
4390 A$=""
4400 DSKIN A$
4410 NEXT
4420 GOTO 4280  'CONTINUE FOR ENTIRE FILE
4430 RESUME 4440  'END OF FILE ERROR
4440 IF LNAME$=TEMP$(1) THEN 4660  'SEE IF LAST RECORD WRITTEN
4450 IF TEMP$(1)="ZZZZZ" THEN 4660
4460 DSKOUT FC$  'WRITE RECORD FUNCTION
4470 DSKOUT TEMP$(1)  'WRITE RECORD NAME
4480 LNAME$=TEMP$(1)  'STORE NAME JUST WRITTEN
4490 TEMP$(1)="ZZZZZ"  'STORE MARKER TO START AGAIN
4500 IF FC$="DEL" THEN 4540  'SKIP REST OF RECORD IF DELETE
4510 FOR I=2 TO HNUM  'WRITE REST OF RECORD OTHERWISE
4520 DSKOUT TEMP$(I)
4530 NEXT
4540 CLOSR  'REINITIALIZE SCR1 TO START AGAIN
4550 OPENR SCR$
4560 GOTO 4280  'CONTINUE SEARCH
4570 CLOSR  'ALL RECORDS WERE IN ALPHABETICAL ORDER SO COPY FILE TO UPDA
4580 OPENR SCR$
4590 OPENW UP$
4600 ON ERROR GOTO 4650  'END OF FILE ERROR
4610 A$=""
4620 DSKIN A$  'READ FROM SCR1 AND COPY TO UPDATE FILE
```

INFO CONTINUED

```
4630 DSKOUT A$
4640 GOTO 4610
4650 RESUME 4660   END OF FILE ROUTINE
4660 CLOSR
4670 CLOSW
4680 OPENR UP$   NOW BEGIN MERGE PROCESS
4690 OPENW SCR$   MERGE OLD DATA FILE AND UPDATE FILE INTO ONE FILE
4700 LNAME$=""
4710 ON ERROR GOTO 5180   END OF FILE ERROR
4720 FUNC$=""
4730 DSKIN FUNC$   READ FUNCTION
4740 IF FUNC$ "DEL" THEN 4800   SKIP TO NEXT RECORD IF DELETE
4750 FOR I=1 TO HNUM   READ REST OF RECORD
4760 TEMP$(I)=""
4770 DSKIN TEMP$(I)
4780 NEXT
4790 GOTO 4820   GO CHECK NAME
4800 TEMP$(1)=""   READ NEXT NAME
4810 DSKIN TEMP$(1)
4820 IF TEMP$(1)<=LNAME$ THEN 4710   'SEE IF RECORD ALREADY PROCESSED
4830 CLOSR   'IF NOT SWITCH TO OLD DATA FILE
4840 OPENR FILE$
4850 DSKIN HNUM   SKIP OVER HEADING DATA
4860 FOR I=1 TO HNUM+1
4870 NAME$=""
4880 ON ERROR GOTO 5210   END OF FILE ERROR
4890 DSKIN NAME$
4900 NEXT
4910 IF NAME$<TEMP$(1) THEN 5000   'GO COPY ALL RECORDS LESS THAN CURRENT
4915 REM PROCESS CURRENT RECORD
4920 IF FUNC$ "DEL" THEN 4960   SKIP DELETIONS
4930 FOR I=1 TO HNUM   'COPY OTHER DATA
4940 DSKOUT TEMP$(I)
4950 NEXT
4960 LNAME$=TEMP$(1)   STORE NAME OF CURRENT RECORD
4970 CLOSR   'SWITCH BACK TO UPDATE FILE FOR NEXT RECORD
4980 OPENR UP$
4990 GOTO 4710   GO BACK AND CONTINUE
5000 IF NAME$<LNAME$ THEN 5070   'SEE IF THIS NAME NEEDS TO BE WRITTEN
5010 FOR I=1 TO HNUM   SKIP RECORD
5020 NAME$=""
5030 ON ERROR GOTO 5210   END OF FILE ERROR
5040 DSKIN NAME$   GET NAME OF NEXT RECORD
5050 NEXT
5060 GOTO 4910   GO BACK AND CHECK IT
5070 DSKOUT NAME$   'COPY RECORD TO SCR1
5080 LNAME$=NAME$   'STORE LAST NAME WRITTEN
5090 FOR I=2 TO HNUM   'COPY REST OF RECORD
5100 A$=""
5110 DSKIN A$
5120 DSKOUT A$
5130 NEXT
5140 ON ERROR GOTO 5210   'END OF FILE ERROR
5150 NAME$=""
5160 DSKIN NAME$   'GET NAME OF NEXT RECORD
5170 GOTO 4910   'GO BACK AND CHECK IT
5180 RESUME 5190   'END OF UPDATE FILE
5190 TEMP$(1)="ZZZZZ"   'SET VALUE TO INDICATE EOF
5200 GOTO 4830   'CONTINUE COPYING DATA FILE
5210 RESUME 5220   'END OF DATA FILE
5220 IF TEMP$(1)="ZZZZZ" THEN 5440   'CHECK IF UPDATE FILE EXHAUSTED
5230 CLOSR
5240 OPENR UP$   'IF NOT THEN COPY REST OF UPDATE FILE
5250 ON ERROR GOTO 5430   END OF FILE ERROR
5260 FUNC$=""
5270 DSKIN FUNC$   'GET NEXT FUNCTION
5280 IF FUNC$="DEL" THEN 5340   'SHOULD NOT BE ANY DELETIONS
5290 FOR I=1 TO HNUM   'READ REST OF RECORD
5300 TEMP$(I)=""
5310 DSKIN TEMP$(I)
5320 NEXT
5330 GOTO 5370   'GO CHECK IF IT IS WRITTEN ALREADY
5340 TEMP$(1)=""
5350 DSKIN TEMP$(1)   'SAFETY MEASURE IN CASE DELETION OCCURS
5360 GOTO 5250   'READ IN JUST NEXT NAME AND CONTINUE
5370 IF TEMP$(1)<=LNAME$ THEN 5250   'SEE IF ALREADY WRITTEN
5380 FOR I=1 TO HNUM   'COPY RECORD IF NOT
5390 DSKOUT TEMP$(I)
5400 NEXT
5410 LNAME$=TEMP$(1)   'STORE NAME JUST WRITTEN
5420 GOTO 5250   'CONTINUE COPYING
5430 RESUME 5440   'END OF FILE ERROR
5440 CLOSW
5450 CLOSR
5455 REM NOW COPY UPDATED FILE BACK INTO DATA FILE
5460 OPENR SCR$
5470 OPENW FILE$
5480 DSKOUT HNUM   'STORE NUMBER OF HEADINGS
5490 FOR I=1 TO HNUM   'STORE NAMES OF HEADINGS
5500 DSKOUT HINAME$(I)
5510 NEXT
5520 ON ERROR GOTO 5570   'END OF FILE ERROR
5530 A$=""
5540 DSKIN A$
5550 DSKOUT A$
5560 GOTO 5530   'CONTINUE COPYING FILE
5570 RESUME 5580   'END OF FILE ROUTINE
5580 PRINT
5590 CLOSW   'CLOSE ALL FILES
5600 CLOSR
5610 FOR I=1 TO 10   'OUTPUT FINISHED MESSAGE
5620 PRINT"INFO IS NOW FINISHED!!"
5630 PRINT
5640 NEXT
5650 END
```

SEARC

Another option for this program would be the comparison of two heading items for each entry. For example, if one heading contains the amount of a chemical (or inventory item) on hand and another heading contains the minimum amount which is desirable, a comparison of these data items could produce a list of chemicals which need to be ordered at any given time. This could be accomplished by having the user enter the heading numbers and having the program compare the values for the respective headings.

Another modification which might be helpful is to permit numerical values within a data field to be treated as numbers rather than character strings. This could be done by modifying line numbers 2744-2820 and 1480 to ignore non-numeric characters.

If memory size is a problem, the help sequence (line numbers 2830-3080) can be omitted and the program could be split into several smaller programs which only permit one or two search options.

The following list contains all the variables used in SEARC and the function each variable serves.

Name	Description
A$	used to input responses
B$(SIZE)	stores standards for each search (if needed)
C	flag set to 1 for ALL type search, set to 2 for ANY type search
CASE	flag used to indicate whether the program is just starting or has already done part of its work
CC	counts number of lines printed on the current page
CRT	number of printable lines minus 2 on a television or cathode ray type terminal
D$	inputs date
D1$	can store the type of file being searched (such as CHEMICALS)
E$(HNUM)	stores all data for the entry currently being processed
FILE$	stores name of the data file and drive number
H(SIZE)	stores heading numbers to be searched for each criterion
HNUM	stores the number of headings in the data file
I	loop parameter
I1	loop parameter
J	loop parameter
K6(3)	used in searching for numeric data
K8	flag used in searching for numeric data
LR	stores the length of R$
M$	used to input responses to system messages
M3	stores the number of lines printed

M4	stores the number of copies to be printed
M5	the number of printable lines per page on a paper-printing terminal
M7	the number of copies currently printed
M9	length of a column (when data is printed in column form) in characters
N	used to input heading numbers for search criteria
N2	counter used in tabbing
N3	width of each column for uniform spacing (when data is printed in column form) in characters
N4	flag to indicate whether column headings need to be printed
N5	flag to suppress or cause heading printout; counts the number of headings being searched
N6	counts the number of entries which satisfy the search criteria
N7	flag to indicate column or row format for output
N8	counts the number of entries in the file
NN	stores the number of characters/line on the terminal or printer
PAPER	number of characters per line on paper-printing terminal
PSIZE	number of lines per page on paper-printing terminal or printer
R$	stores legal numerical characters
R(SIZE)	stores the search type number for each criterion
S$(HNUM)	stores the names of the headings in the data file
S2	loop parameter
SIZE	maximum number of search criteria allowed
SKIP	loop parameter
TV	number of characters per line on television or screen type terminal
V(HNUM)	stores flags to print headings if needed
V1	loop parameter
V2	parameter to indicate which heading is currently being printed
V4	flag to indicate whether a numerical search was requested (value of 1 if so)
V5	loop parameter
V6	loop parameter
V7	flag to indicate that a heading needs to be printed
W(SIZE)	computes the sum for numerical searches
Y$	used to input responses to system messages
Z$	used to input responses to system messages

SEVC LISTING

```
1 REM *** PROGRAMMED BY VIK MUIZNIEKS 11/77
2 REM ***
3 PRINT "ALWAYS PRESS THE RETURN KEY AFTER TYPING-IN A RESPONSE!!!!"
5 CLEAR 500  'SET UP STRING STORAGE SPACE
10 GOSUB 3200  'GET FILE NAME
11 REM *** READ MAX SEARCH SIZE, TERMINAL PARAMETERS, FILE TITLE
12 REM *** SIZE=MAX # OF SEARCH SPECIFICATIONS/SEARCH
13 REM *** TV=# OF CHAR'S PER LINE ON CRT SCREEN
14 REM *** PAPER=# OF CHAR'S PER LINE ON PAPER-PRINTING TERMINAL
15 REM *** M5=# OF PRINTABLE LINES PER PAGE ON PAPER-PRINTING TERMINAL
16 REM *** CRT=# OF DISPLAYABLE LINES ON CRT SCREEN MINUS TWO
17 REM *** D1$=TITLE OF DATA FILE BEING SEARCHED
18 READ SIZE,TV,PAPER,M5,CRT,D1$
20 R$=" 0123456789.+-"  'LEGAL CHARACTERS IN NUMERIC STRING
21 LR=LEN(R$)  'GET LENGTH OF NUMBER CHECKING STRING
25 DIM K6(3)  'USED IN FINDING NUMBERS
26 REM *** GET INFO ABOUT PRINTOUT
27 PRINT "ARE YOU USING A PAPER-PRINTING TERMINAL (YES OR NO)";
28 Y$="": INPUT Y$
29 IF LEFT$(Y$,1)="Y" THEN 33
30 IF LEFT$(Y$,1)<>"N" THEN 27
31 NN=TV: PSIZE=CRT: M4=1: V7=1: WIDTH TV 'SET TO CRT PARAMETERS FOR
32 GOTO 40
33 NN=PAPER: M4=0: V7=0: PSIZE=0: WIDTH PAPER 'SET TO PAPER-PARAMETERS
34 PRINT "WHAT IS TODAY'S DATE (NO COMMAS PLEASE)";
35 D$="": INPUT D$
40 CASE=0  'NO REINITIALIZATION
61 PRINT "DO YOU NEED HELP (YES OR NO)";
62 Y$="": INPUT Y$
63 IF LEFT$(Y$,1)="N" THEN 80
64 IF LEFT$(Y$,1)<>"Y" THEN 61
65 GOSUB 2830  'PRINT OUT HELP SEQUENCE
80 IF CASE=0 THEN 100
85 ERASE S$,E$
90 CLOSR
100 ON ERROR GOTO 103
101 OPENR FILE$
102 GOTO 120
103 RESUME 104
104 CLOSR
105 PRINT "DATA FILE IS NOT ON DRIVE 1.  PUT DISKETTE WITH DATA FILE"
106 PRINT "IN DRIVE 1 AND PRESS THE RETURN KEY ON THE TERMINAL."
107 A$="": INPUT A$
108 GOTO 100
120 HNUM=0: DSKIN HNUM 'GET # OF HEADINGS
130 DIM S$(HNUM),E$(HNUM)
135 IF CASE=0 THEN DIM V(HNUM) 'ONLY INITIALIZE ARRAY SPACE ONCE
140 FOR I=1 TO HNUM  'GET HEADING NAMES
150 S$(I)="":DSKIN S$(I)
160 NEXT I
165 IF CASE=2 THEN 2338  'REINITIALIZING FILES
170 PRINT 'DO YOU WANT A LIST OF HEADINGS (YES OR NO)";
180 A$="":INPUT A$
190 IF LEFT$(A$,1)="N" THEN 250
200 IF LEFT$(A$,1)<>"Y" THEN 170
210 PRINT "HERE ARE THE HEADINGS AND THEIR NUMBERS"
215 REM *** PRINT LIST OF HEADINGS
216 CC=0 'LINES PRINTED COUNTER
220 FOR I=1 TO HNUM
230 PRINT I,")  ";S$(I)
231 CC=CC+1  'INCREMENT LINES-PRINTED COUNTER
232 IF CC<CRT THEN 240  'FULL PAGE YET?
233 PRINT "TYPE JUST THE RETURN KEY FOR MORE.";
234 A$="":INPUT A$
235 IF A$<>"" THEN 250  'WANT MORE?
236 CC=0  'RESET LINES-PRINTED COUNTER
240 NEXT I
250 PRINT "DO YOU NEED A LIST OF SEARCH TYPES (YES OR NO)";
260 A$="":INPUT A$
270 IF LEFT$(A$,1)="N" THEN 350
280 IF LEFT$(A$,1)<>"Y" THEN 250
290 PRINT
300 PRINT "CHOICES:  1) EXACT MATCH, 2)BEGINNING MATCHES STANDARD"
310 PRINT "3) <= STANDARD, 4) >= STANDARD, 5) LOOK FOR NUMBERS"
320 PRINT "6) LOOK FOR STANDARD ANYWHERE WITHIN THE FIELD"
330 PRINT
350 REM *** SET UP INITIAL PARAMETER VALUES
360 N4=0: N5=0: N6=0: N7=0: N8=0: V4=0: M3=0: M7=0
380 PRINT
720 PRINT "SPECIFY THE HEADING NUMBER(S) AND TYPE(S) OF SEARCHES"
730 PRINT "THAT YOU WANT; ENTER 0 FOR HEADING NUMBER WHEN DONE"
733 IF CASE<>0 THEN 735
734 DIM H(SIZE),B$(SIZE),R(SIZE),W(SIZE)
735 FOR I=1 TO SIZE  'INIT HEADING #'S TO BE SEARCHED
736 H(I)=0
737 NEXT I
739 REM *** GET HEADINGS TO BE SEARCHED
740 FOR I1=1 TO SIZE
741 IF I1=1 THEN 750
742 PRINT "TYPE A HEADING NUMBER TO SPECIFY ANOTHER SEARCH CONDITION,"
743 PRINT "TYPE 0 IF YOU ARE DONE, OR -1 TO START ALL OVER."
744 CASE=-10
```

SEARC CONTINUED

```
750 PRINT
760 PRINT "HEADING NUMBER";
770 H(I1)=-2: INPUT H(I1)
780 IF H(I1)=0 THEN 960  'SEE IF DONE ENTERING HEADINGS
785 IF H(I1)=-1 THEN 360  'RESET V4 VALUE AND TRY AGAIN
790 IF H(I1)<1 OR H(I1)>HNUM THEN 930
795 REM *** GET CORRESPONDING SEARCH TYPES
810 PRINT "SEARCH TYPE NUMBER";
820 R(I1)=0: INPUT R(I1)
830 IF R(I1)<1 OR R(I1)>6 THEN 910
850 IF R(I1) <>5 THEN 880  'SEE IF WANT #'S, IF SO, NO STANDARD
860 V4=1  'INDICATE AT LEAST ONE COLUMN TOTAL REQUESTED
870 GOTO 950
875 REM *** GET STANDARD FOR SEARCH
880 PRINT "STANDARD";
890 B$(I1)="": INPUT B$(I1)
900 GOTO 950
910 PRINT "CHOICES ARE 1-6, TRY AGAIN"
920 GOTO 820
930 PRINT "CHOICES ARE 1-";HNUM;", TRY AGAIN"
940 GOTO 770
950 NEXT I1
960 IF H(1)=0 THEN 2290  'SEE IF NO HEADINGS REQUESTED
961 C=1  'ALL CONDITIONS MUST BE MET BY QUALIFYING ITEM
970 I1=I1-1
980 IF I1=1 THEN 1050  'ONLY ONE HEADING SPECIFIED?
990 PRINT  'SEE IF WANT ALL OR ANY SEARCH TYPE
1000 PRINT
1010 PRINT "1 FOR ALL CONDITIONS TO BE SATISFIED, 2 IF ANY WILL SUFFICE"
1020 C=0: INPUT C
1030 IF C<1 OR C>2 THEN 1010
1050 GOSUB 2450  'GET WHICH HEADINGS ARE TO BE LISTED
1060 GOSUB 2120  'FIND OUT IF WANT COLUMN OR ROW FORMAT
1070 REM *** NOW BEGIN SEARCHING THROUGH FILE
1100 ON ERROR GOTO 2190  'READ TILL END-OF-FILE ENCOUNTERED
1110 FOR S2=1 TO HNUM 'READ ALL DATA FOR ENTRY
1120 E$(S2)="": DSKIN E$(S2)
1150 NEXT S2
1170 N8=N8+1  'INCREMENT COUNTER ON # OF ENTRIES IN FILE
1180 REM *** DETERMINE THE TYPE OF SEARCH BEING CONDUCTED
1190 FOR I=1 TO I1
1200 ON R(I) GOTO 1220,1250,1280,1310,2740,1231
1210 REM *** LOOK FOR EXACT MATCH WITH STANDARD
1220 IF B$(I)=E$(H(I)) THEN 1360  'ITEM MEETS CRITERIA
1230 GOTO 1320  'ITEM FAILS TO MEET CRITERIA
1231 REM *** LOOK FOR SUBSTRING
1232 IF INSTR(E$(H(I)),B$(I))=0 THEN 1320 ELSE 1360
1240 REM *** LOOK FOR SAME BEGINNING ON LEFT SIDE AS STANDARD
1250 IF B$(I)=MID$(E$(H(I)),1,LEN(B$(I))) THEN 1360
1260 GOTO 1320
1270 REM *** SEE IF DATA ITEM IS <= STANDARD
1280 IF B$(I)>=E$(H(I)) THEN 1360
1290 GOTO 1320
1300 REM *** SEE IF DATA ITEM IS >= STANDARD
1310 IF B$(I)<=E$(H(I)) THEN 1360
1315 REM *** SEE IF DOING AN ALL OR ANY SEARCH
1320 IF C=1 THEN 1100  'ITEM DOES NOT MEET REQUIREMENTS
1330 IF I=I1 THEN 1100  'ALL SEARCH TYPES EXHAUSTED
1340 GOTO 1520  'CONTINUE EXAMINING SINCE ALL SEARCH
1360 IF C=1 THEN 1510  'SEE IF DOING AN ALL SEARCH
1370 REM COMPUTE COLUMN TOTALS FOR ITEMS MEETING SEARCH CRITERIA
1380 IF I=I1 THEN 1540  'IF DONE, GO PRINT ITEM
1390 FOR J=I+1 TO I1
1400 IF R(J)<>5 THEN 1490  'SEE IF GETTING COLUMN TOTALS
1480 W(J)=W(J)+VAL(E$(H(J)))  'INCREASE COL. TOTAL, NON-#'S ADD 0
1490 NEXT J
1500 GOTO 1540  'PRINT ITEM
1505 REM *** SEE IF ALL CRITERIA MET
1510 IF I=I1 THEN 1540  'IF DONE, GO PRINT ITEM
1520 NEXT I
1530 REM *** DETERMINE WHAT INFORMATION TO PRINT OUT
1540 IF V7=0 THEN 1640  'PRINT HEADING FOR REPORT IF NOT DONE
1550 IF N7=1 THEN 1800  'DETERMINE IF ROW OR COLUMN FORM
1560 REM *** PRINT OUT ROW FORM
1570 FOR S2=1 TO HNUM
1580 IF V(S2)=0 THEN 1600  'DOES HEADING NEED TO BE PRINTED?
1590 PRINT S$(S2);" = ";E$(S2)  'PRINT HEADING NAME AND DATA
1591 M3=M3+1  'INCREASE # OF LINES PRINTED
1592 GOSUB 3900  'CHECK IF PAGE FULL
1600 NEXT S2
1610 N6=N6+1  'INCREMENT COUNTER ON # OF ITEMS FOUND
1620 PRINT
1621 M3=M3+1
1622 GOSUB 3900  'CHECK IF PAGE FULL
1630 GOTO 1100  'GET NEXT DATA ITEM IN FILE
1639 REM *** GET INFO ABOUT PRINTOUT
1640 IF M4>0 AND N5<>0 THEN 1750  'ALREADY GOT THIS INFORMATION
1641 IF M4>0 AND N5=0 THEN 1780  'SKIP OVER PRINTING HEADING
1642 PRINT "HOW MANY PRINTED COPIES OF THE SEARCH RESULTS DO YOU WANT";
1643 M4=0:  INPUT M4
1644 IF M4<=0 THEN 1642
1645 PRINT "ROLL PAPER TO THE TOP OF THE PAGE AND PRESS RETURN ";
1646 M$="":  INPUT M$
1647 IF N5=0 THEN 1780
1745 REM *** PRINTOUT LISTING HEADING
1750 PRINT
1760 PRINT FILE$;" FILE LISTED ON ";D$  'PRINT OUT NAME OF FILE
1770 PRINT
```

SEARC CONTINUED

```
1771 M3=M3+3
1772 GOSUB 3900  'CHECK IF PAGE FULL
1780 V7=1  'INDICATE HEADING INFORMATION PRINTED
1790 IF N7=0 THEN 1570  'PRINT OUT ROW FORM IF NECESSARY
1800 IF N4=1 THEN 2010  'ARE COLUMN HEADINGS PRINTED?
1810 N4=1  'INDICATE COLUMN HEADINGS PRINTED
1820 REM *** SEE IF NEED TO PRINT OUT HEADING NAMES
1830 IF N5<>0 THEN 1850  'SEE IF NOT PRINTING ANY DATA
1840 GOTO 2090  'IF NOT, GO AWAY
1850 M7=M7+1  'INCREASE # OF COPIES PRINTED COUNTER
1851 IF M7<>1 THEN 1890 'ONLY SET PAGE WIDTH 1ST TIME THROUGH
1855 N3=INT(NN/N5)  'DETERMINE WIDTH OF EACH COLUMN-UNIFORM SPACING
1890 N2=0  'INITIALIZE TABBING COUNTERS FOR PRINTING
1900 V2=1
1910 IF V(V2)=0 THEN 1950  'SEE IF HEADING IS TO BE PRINTED
1920 IF S$(V2)="" THEN 1940
1930 PRINT TAB(N3*N2);MID$(S$(V2),1,N3);
1940 N2=N2+1  'INCREMENT FIELD POINTER FOR TABBING
1950 V2=V2+1  'GET NEXT COLUMN DATA
1960 IF N2=N5 THEN 1980  'PRINTED ALL COLUMNS?
1970 GOTO 1910  'PRINT NEXT COLUMN
1980 PRINT
1990 PRINT
1991 M3=M3+2
1992 GOSUB 3900  'CHECK IF PAGE FULL
2000 REM *** PRINT OUT DATA ITEMS UNDER PROPER COLUMN HEADINGS
2010 N2=0  'SET FIELD COUNTER FOR PRINTING
2015 IF N5=0 THEN 2090  'SEE IF NOT PRINTING ANY DATA
2020 FOR S2=1 TO HNUM
2030 IF V(S2)=0 THEN 2070  'IGNORE NON-PRINTING FIELDS
2040 IF E$(S2)="" THEN 2060  'AND ENTIRELY BLANK FIELDS
2041 M9=LEN(E$(S2))  'SEARCH TO SEE IF STRING OF BLANKS
2042 FOR M8=1 TO M9  'IF SO, DON'T PRINT THEM
2043 IF MID$(E$(S2),M8,1)<>" " THEN 2050
2044 NEXT M8
2045 GOTO 2060
2050 PRINT TAB(N3*N2);MID$(E$(S2),1,N3-1);
2060 N2=N2+1  'INCREMENT FIELD PRINTING POINTER
2070 NEXT S2
2080 PRINT
2081 M3=M3+1  'INCREMENT NUMBER OF LINES PRINTED COUNTER
2082 GOSUB 3900  'CHECK IF PAGE FULL
2090 N6=N6+1  'INCREMENT COUNTER ON NUMBER OF MATCHING ITEMS FOUND
2100 GOTO 1100
2110 REM *** PRINT OUT TOTAL INFORMATION FOR SEARCH AND COLUMNS
2120 IF N5=0 THEN 2170  'SEE IF NO HEADINGS ARE TO BE LISTED
2122 PRINT "ARE THE RESULTS TO BE LISTED IN COLUMN FORM (YES OR NO)";
2130 Z$="": INPUT Z$
2140 IF LEFT$(Z$,1)="Y" THEN 2170
```



```
2150 IF LEFT$(Z$,1)<>"N" THEN 2122
2160 GOTO 2180  'DEFAULT IS ROW FORM, ALREADY 0
2170 N7=1  'FLAG TO PRINTOUT IN COLUMN FORM
2180 RETURN
2185 REM *** DONE PRINTING SPECIFIC ITEMS
2190 RESUME 2195
2191 IF PSIZE=0 OR M3<PSIZE-3 THEN 2195
2192 M3=PSIZE+1  'FUDGE TERMINAL SIZE TO GET ANOTHER PAGE
2193 GOSUB 3900  'CHECK IF PAGE FULL--WHICH IT IS
2195 PRINT
2200 PRINT
2210 PRINT N6;" OF THE ";N8;" ITEMS IN THE FILE MEET THE REQUIREMENTS"
2220 PRINT
2230 PRINT
2231 M3=M3+3
2240 IF V4=0 THEN 2330
2250 FOR V5=1 TO I1
2260 IF R(V5)<>5 THEN 2280
2270 PRINT "TOTALS UNDER HEADING ";S$(H(V5));" = ";W(V5)
2271 IF N6<>0 THEN PRINT "   AVERAGE OF VALUES FOUND = ";W(V5)/N6
2272 IF N6<>0 THEN M3=M3+2 ELSE M3=M3+1
2273 GOSUB 3900  'SEE IF PAGE FULL
2280 NEXT V5
2290 PRINT
2320 PRINT
2325 M3=M3+2
2326 GOSUB 3900  'CHECK IF PAGE FULL
2330 M4=M4-1  'DECREASE NUMBER OF COPIES NEEDED
2331 FOR I=1 TO SIZE: W(I)=0  'REINITIALIZE TOTALS ARRAY FOR NEXT SEARCH
2332 NEXT I
2333 CASE=2 'REINITIALIZE DATA FILE
2334 GOTO 80
2338 REM *** RESTORE NECESSARY DATA VALUES FOR NEXT PRINTOUT
2339 IF PSIZE=0 THEN GOSUB 3970: IF PSIZE=0 THEN V7=0: M3=0: N4=0: N6=0: N8=0
2340 IF M4>=1 THEN 1070  'REPEAT SEARCH IF NECESSARY
2341 PRINT "DO YOU WANT TO PERFORM ANOTHER SEARCH (YES OR NO)";
2342 A$="": INPUT A$
2343 IF LEFT$(A$,1)="N" THEN 4000
2344 IF LEFT$(A$,1)<>"Y" THEN 2341
2345 CASE=1: IF PSIZE <>0 THEN M4=1
2346 GOTO 61  'PREPARE FOR A COMPLETELY NEW SEARCH
2348 REM *** SUB TO INITIALIZE HEADINGS AND TOTALS TO BE LISTED
2450 FOR V1=1 TO SIZE
2470 W(V1)=0  'AND TO SET COLUMN TOTALS TO ZERO
2480 NEXT V1
2481 FOR V1=1 TO HNUM
2482 V(V1)=0  'DEFAULT IS NOT TO LIST ANY HEADINGS
2483 NEXT V1
2490 PRINT
```

SEARC CONTINUED

```
2500 PRINT
2510 PRINT "NOW SPECIFY THE NUMBERS OF ALL THE HEADINGS YOU WANT"
2520 PRINT "PRINTED.  ENTER 0 WHEN DONE, OR -1 FOR ALL THE HEADINGS."
2530 FOR V1=1 TO SIZE
2540 N=-2: INPUT N
2550 IF N=-1 THEN 2660  'WANT ALL HEADINGS?
2560 IF N=0 THEN 2700  'DONE SPECIFYING HEADINGS
2570 IF N<1 OR N>HNUM THEN 2630
2590 IF V(N)=1 THEN 2640  'HEADING ALREADY PRESENT?
2600 V(N)=1  'IF NOT, ADD IT
2610 N5=N5+1  'INCREASE COUNT ON # OF HEADINGS TO BE LISTED
2620 GOTO 2640  'GET NEXT HEADING #
2630 PRINT "ENTER ONLY -1, 0, OR A NUMBER FROM 1 TO ";HNUM
2640 NEXT V1
2650 GOTO 2700
2655 REM *** INDICATE ALL HEADINGS TO BE LISTED
2660 FOR V1=1 TO HNUM
2670 V(V1)=1  'LIST HEADING
2680 NEXT V1
2690 N5=HNUM  'SET # OF HEADINGS TO BE PRINTED TO MAXIMUM
2700 RETURN
2730 REM *** SEE IF DATA ITEM IS NUMBER, UPDATE COLUMN TOTAL
2740 K8=0  'FLAG TO SEE IF NUMBER PRESENT
2741 FOR V6=1 TO 3  'CLEAR ARRAY FOR FINDING MULTIPLE + .  -
2742 K6(V6)=0
2743 NEXT V6
2744 FOR V6=1 TO LEN(E$(H(I)))
2750 FOR V5=1 TO LR  'SCAN THROUGH ALL ACCEPTABLE NUMBER SYMBOLS
2760 IF MID$(E$(H(I)),V6,1)=MID$(R$,V5,1) THEN 2790
2770 NEXT V5
2780 GOTO 1320  'NO NUMBER
2790 IF V5=1 THEN 2795
2791 IF V5<12 THEN 2794  'CHECK FOR + .  OR -
2792 IF K6(15-V5)=1 THEN 1320 ELSE K6(15-V5)=1  'RETURN IF NO NUM.
2793 GOTO 2795  'NO GOOD CHARACTER YET
2794 K8=2  'INDICATE NUMERAL FOUND
2795 NEXT V6
2800 IF E$(H(I))="" THEN 1320  'NULL STRING NOT COUNTED AS A NUMBER
2805 IF K8=0 THEN 1320  'NEITHER IS A STRING OF BLANKS/SPECIAL SYMBOLS
2810 W(I)=W(I)+VAL(E$(H(I)))
2820 GOTO 1360
2830 REM *** HELP SEQUENCE
2850 PRINT "YOU WILL BE ASKED TO SPECIFY THE HEADING NUMBERS OF ALL THE"
2855 PRINT "HEADINGS YOU WANT TO SEARCH UNDER, AND THE PARTICULAR TYPE"
2860 PRINT "OF SEARCH YOU WANT TO CONDUCT.  THE SEARCHES CAN BE OF"
2865 PRINT "SIX TYPES:  1) LOOK FOR ALL EXACT MATCHES TO A SPECIFIED"
2870 PRINT "STANDARD UNDER THAT HEADING, 2) LOOK FOR ALL ITEMS UNDER"
2875 PRINT "THE HEADING THAT HAVE THE SAME FIRST LETTERS AS THE"
2880 PRINT "SPECIFIED STANDARD, 3) LOOK FOR ALL ITEMS UNDER THE"
2885 PRINT "HEADING THAT HAVE A VALUE LESS THAN OR EQUAL TO A SPECIFIED"
2890 PRINT "STANDARD, 4) LOOK FOR ALL ITEMS UNDER THE HEADING THAT HAVE"
2895 PRINT "A VALUE GREATER THAN OR EQUAL TO A SPECIFIED STANDARD,"
2900 PRINT "5) LOOK FOR ALL ITEMS UNDER THE HEADING THAT HAVE A"
2905 PRINT "NUMERICAL EQUIVALENT; BE CAREFUL, IF AN ENTRY CONTAINS"
2910 PRINT "2 NUMBERS WITH BLANKS IN BETWEEN, THEY WILL BE TREATED AS"
2915 PRINT "ONE NUMBER--E.G.  '12 34' WOULD BE INTERPRETED AS '1234'."
2920 PRINT "AS A SIDE BONUS, YOU GET THE TOTAL FOR ALL OF THE NUMBERS"
2925 PRINT "FOUND UNDER THE HEADING THAT MEET THE REST OF THE SEARCH"
2930 PRINT "CRITERIA (IF ANY).  6) LOOK FOR A PARTICULAR STANDARD"
2935 PRINT "ANYWHERE WITHIN THE FIELD--E.G.  IF THE DATA WERE 'ABEL'"
2940 PRINT "'BEETOVEN' 'BACH'  'JEZABEL'  A STANDARD OF 'BE' WOULD"
2945 PRINT "FIND 'ABEL' 'BEETOVEN' AND 'JEZABEL' BUT NOT 'BACH'."
2946 PRINT
2947 PRINT"PRESS RETURN FOR MORE HELP, ANYTHING ELSE TO GO BACK.";
2948 Y$="": INPUT Y$
2949 IF Y$<>"" THEN RETURN
2950 PRINT "  YOU MAY ASK FOR A HEADING TO BE SEARCHED IN MORE"
2955 PRINT "THAN ONE WAY.  HOWEVER, A MAXIMUM OF ";SIZE;" SEARCH"
2960 PRINT "HEADINGS HAS BEEN IMPOSED.  ONCE YOU HAVE SPECIFIED"
2965 PRINT "ALL OF THE HEADINGS YOU WANT SEARCHED, THEIR CORRESPON-"
2970 PRINT "DING SEARCH TYPES, AND THEIR RELATED STANDARDS (IF"
2975 PRINT "NECESSARY), YOU MUST ALSO INDICATE 1) IF ALL OF THE"
2980 PRINT "CONDITIONS SPECIFIED MUST BE MET BEFORE AN ITEM IS"
2985 PRINT "INCLUDED (I.E.  AN 'AND' SEARCH), OR IF 2) MEETING ANY"
2990 PRINT "OF THE CRITERIA IS ENOUGH TO INCLUDE AN ITEM (I.E.  AN"
2995 PRINT "'OR' SEARCH TYPE) (UNLESS YOU ARE ONLY SEARCHING UNDER"
3000 PRINT "ONE HEADING, IN WHICH CASE IT DOESN'T MATTER).  ALSO,"
3005 PRINT "YOU ARE GIVEN THE CHOICE AS TO WHICH HEADINGS TO HAVE"
3010 PRINT "DISPLAYED FOR THE ITEMS THAT MATCH THE SEARCH CRITERIA"
3015 PRINT "(I.E.  THE PRINTED HEADINGS NEED NOT BE THE SAME AS THE"
3020 PRINT "SPECIFIED HEADINGS IN THE SEARCH)."
3021 PRINT
3022 PRINT "PRESS RETURN FOR MORE HELP, ANYTHING ELSE TO GO BACK.";
3023 Y$="": INPUT Y$
3024 IF Y$<>"" THEN RETURN
3025 PRINT "THE PROGRAM WILL ALSO ASK YOU IF"
3026 PRINT "YOU WANT THE RESULTS LISTED IN A TABULAR"
3030 PRINT "(COLUMN) FORM, AS OPPOSED TO A ROW (LINE BY LINE) FORM."
3035 PRINT "YOU WILL BE ASKED WHETHER OR NOT YOU ARE USING A PAPER-"
3040 PRINT "PRINTING TERMINAL; IF YOU ARE, NO PAGING WILL BE DONE."
3045 PRINT "IF YOU ARE USING A TV-LIKE TERMINAL, PAGING WILL OCCUR"
3050 PRINT "(I.E.  ONLY ONE SCREEN-FULL OF INFORMATION WILL BE SENT"
3055 PRINT "TO YOU AT A TIME; YOU CAN CHOOSE IF YOU WANT MORE SENT)."
3060 PRINT "YOU HAVE THE OPTION OF SEARCHING THROUGH THE FILE AS"
3065 PRINT "MANY TIMES AS YOU WANT TO, AND OF GETTING AS MANY"
3070 PRINT "COPIES OF THE PRINTED RESULTS AS YOU LIKE."
3075 PRINT
3080 RETURN
3170 GOTO 4000
```

SEARC CONTINUED

```
3200 REM *** GET FILE NAME OF DATA FILE
3210 PRINT "WHAT IS THE NAME OF THE DATA FILE IN DRIVE 1"
3230 FILE$="": INPUT FILE$
3240 IF FILE$="" THEN 3210
3250 FILE$=FILE$+": 1"
3260 RETURN
3900 REM *** SEE IF TERMINAL SCREEN FULL
3910 IF PSIZE=0 OR M3<PSIZE THEN RETURN
3920 M3=0  'RESET PAGE LINE COUNTER
3930 PRINT "PRESS RETURN TO CONTINUE, ANYTHING ELSE TO STOP";
3940 Y$="": INPUT Y$
3950 IF Y$="" THEN RETURN
3960 GOTO 2330  'OTHERWISE RESTART NEW SEARCH
3970 REM *** START AT TOP OF NEW PAGE FOR NEXT PRINTED COPY
3972 IF M5>=M3 THEN 3980  'SEE IF PRINTED MORE THAN ONE PAGE
3974 M3=M3-M5  'SUBTRACT EACH PAGE PRINTED
3976 GOTO 3972 'SEE IF ON LAST PAGE
3980 IF M5=M3 THEN RETURN  'HIT A PAGE BOUNDARY EXACTLY?
3981 FOR SKIP=1 TO M5-M3 'SKIP REST OF PAGE
3985 PRINT
3987 NEXT SKIP
3989 RETURN
3990 REM *** DATA PARAMETERS--CAN BE CHANGED TO FIT THE SYSTEM USED
3991 DATA 50
3992 DATA 80
3993 DATA 132
3994 DATA 66
3995 DATA 22
3996 DATA "ENTRY"
4000 REM *** NORMAL ENDING SEQUENCE
4010 CLOSR
4011 ON ERROR GOTO 4014  'CHECK TO MAKE SURE FDOS ON DRIVE 0
4012 OPENR "EXEC"
4013 CLOSR: GOTO 4020
4014 RESUME 4015
4015 CLOSR
4016 PRINT "PLACE A DISKETTE LABELLED 'PROGRAMS' IN DRIVE 0."
4017 PRINT "THEN PRESS THE RETURN KEY ON THE TERMINAL."
4018 A$="": INPUT A$
4019 GOTO 4011
4020 END
```

PSEAR

The PSEAR program essentially is the same as SEARC except that the output statements use the LPRINT command rather than PRINT to format and send the search results to a printer. If your micro-computer system does not support the LPRINT command or has no separate printer, this program may not be needed. However, the ability to print labels can be a useful feature for preparing mailing or folder labels or labels for other uses. Line number 1624 can be changed to allow fewer or more blank lines between outputs, so different label sizes can be used. The variables used in PSEAR are the same as those used in SEARC.

PSEAR LISTING

```
3 PRINT "ALWAYS PRESS THE RETURN KEY AFTER TYPING-IN A RESPONSE!!!!"
5 CLEAR 500
10 GOSUB 3210
18 READ SIZE,TV,PAPER,M5,CRT,D1$
20 R$=" 0123456789 +-"
21 LR=LEN(R$)
25 DIM K6(3)
33 NN=PAPER: M4=0  V7=0  PSIZE=0. WIDTH PAPER
34 PRINT "WHAT IS TODAY'S DATE (NO COMMAS PLEASE)";
35 D$="": INPUT D$
40 CASE=0
61 PRINT "DO YOU NEED HELP (YES OR NO)";
62 Y$="": INPUT Y$
63 IF LEFT$(Y$,1)="N" THEN 80
64 IF LEFT$(Y$,1)<>"Y" THEN 61
65 GOSUB 2850
80 IF CASE=0 THEN 100
85 ERASE S$,E$
90 CLOSR
100 ON ERROR GOTO 103
101 OPENR FILE$
102 GOTO 120
103 RESUME 104
104 CLOSR
105 PRINT "DATA FILE IS NOT ON DRIVE 1.  PUT DISKETTE WITH DATA FILE"
106 PRINT "IN DRIVE 1 AND PRESS THE RETURN KEY ON THE TERMINAL."
107 A$="": INPUT A$
108 GOTO 100
120 HNUM=0: DSKIN HNUM
130 DIM S$(HNUM),E$(HNUM)
135 IF CASE=0 THEN DIM V(HNUM)
140 FOR I=1 TO HNUM
150 S$(I)="": DSKIN S$(I)
160 NEXT I
165 IF CASE=2 THEN 2335
170 PRINT "DO YOU WANT A LIST OF HEADINGS (YES OR NO)";
180 A$="": INPUT A$
190 IF LEFT$(A$,1)="N" THEN 250
200 IF LEFT$(A$,1)<>"Y" THEN 170
210 PRINT "HERE ARE THE HEADINGS AND THEIR NUMBERS"
216 CC=0
220 FOR I=1 TO HNUM
230 PRINT I;")  ";S$(I)
231 CC=CC+1
232 IF CC<CRT THEN 240
233 PRINT "TYPE JUST THE RETURN KEY FOR MORE.";
234 A$="": INPUT A$
235 IF A$<>"" THEN 250
236 CC=0
240 NEXT I
250 PRINT "DO YOU NEED A LIST OF SEARCH TYPES (YES OR NO)";
260 A$="": INPUT A$
270 IF LEFT$(A$,1)="N" THEN 360
280 IF LEFT$(A$,1)<>"Y" THEN 250
290 PRINT
300 PRINT "CHOICES:  1) EXACT MATCH, 2) BEGINNING MATCHES STANDARD,"
310 PRINT"           3) <=STANDARD, 4) >=STANDARD, 5) LOOK FOR NUMBERS,"
320 PRINT"           6) LOOK FOR STANDARD ANYWHERE WITHIN THE FIELD"
330 PRINT
360 N4=0: N5=0: N6=0: N7=0: N8=0: V4=0: M3=0: M7=0
380 PRINT
720 PRINT "SPECIFY THE HEADING NUMBER(S) AND TYPE(S) OF SEARCHES"
730 PRINT "THAT YOU WANT; ENTER 0 FOR HEADING NUMBER WHEN DONE"
733 IF CASE<>0 THEN 735
734 DIM H(SIZE),B$(SIZE),R(SIZE),W(SIZE)
735 FOR I=1 TO SIZE
736 H(I)=0
737 NEXT I
740 FOR I1=1 TO SIZE
741 IF I1=1 THEN 750
742 PRINT "TYPE A HEADING NUMBER TO SPECIFY ANOTHER SEARCH CONDITION,"
743 PRINT "TYPE 0 IF YOU ARE DONE, OR -1 TO START ALL OVER."
744 CASE=-10
750 PRINT
760 PRINT "HEADING NUMBER";
770 H(I1)=-2: INPUT H(I1)
780 IF H(I1)=0 THEN 960
785 IF H(I1)=-1 THEN 360
790 IF H(I1)<1 OR H(I1)>HNUM THEN 930
810 PRINT "SEARCH TYPE NUMBER";
820 R(I1)=0: INPUT R(I1)
830 IF R(I1)<1 OR R(I1)>6 THEN 910
850 IF R(I1) <>5 THEN 880
860 V4=1
870 GOTO 950
880 PRINT "STANDARD";
890 B$(I1)="": INPUT B$(I1)
900 GOTO 950
910 PRINT "CHOICES ARE 1-6, TRY AGAIN"
920 GOTO 820
930 PRINT "CHOICES ARE 1-";HNUM;", TRY AGAIN"
940 GOTO 770
950 NEXT I1
960 IF H(1)=0 THEN 2290
961 C=1
970 I1=I1-1
```

PSEUDO CONTINUED

```
980 IF I1=1 THEN 1650
990 PRINT
1000 PRINT
1010 PRINT "1 FOR ALL CONDITIONS TO BE SATISFIED, 2 IF ANY WILL SUFFICE"
1020 C=0 INPUT C
1030 IF C<1 OR C>2 THEN 1010
1050 GOSUB 2150
1060 GOSUB 2120
1100 ON ERROR GOTO 2190
1110 FOR S2=1 TO MNUM
1120 E$(S2)="" DSKIN E$(S2)
1150 NEXT S2
1170 NB=NB+1
1190 FOR I=1 TO I1
1200 ON R(I) GOTO 1220,1250,1280,1310,2740,1732
1220 IF B$(I)=E$(H(I)) THEN 1360
1230 GOTO 1320
1232 IF INSTR(E$(H(I)),B$(I))=0 THEN 1320 ELSE 1360
1250 IF B$(I)=MID$(E$(H(I)),1,LEN(B$(I))) THEN 1360
1260 GOTO 1320
1280 IF B$(I)>E$(H(I)) THEN 1360
1290 GOTO 1320
1310 IF B$(I)<E$(H(I)) THEN 1360
1320 IF C=1 THEN 1100
1330 IF I=I1 THEN 1100
1340 GOTO 1520
1360 IF C=1 THEN 1510
1380 IF I=I1 THEN 1540
1390 FOR J=I+1 TO I1
1400 IF R(J)<>5 THEN 1490
1480 W(J)=W(J)+VAL(E$(H(J)))
1490 NEXT J
1500 GOTO 1540
1510 IF I=I1 THEN 1540
1520 NEXT I
1540 IF V7=0 THEN 1640
1550 IF N7=1 THEN 1800
1570 FOR S2=1 TO MNUM
1580 IF V(S2)=0 THEN 1600
1585 IF N7=2 THEN LPRINT E$(S2) ELSE 1590
1586 GOTO 1591
1590 LPRINT S$(S2);" = ";E$(S2)
1591 M3=M3+1
1592 GOSUB 3910
1600 NEXT S2
1610 MA=MA+1
1620 LPRINT
1621 M3=M3+1
1622 GOSUB 3910
1623 IF N7<>2 THEN 1630
1624 LPRINT:LPRINT:LPRINT:LPRINT
1630 GOTO 1100
1640 IF M4>0 AND N5<>0 THEN 1750
1641 IF M4>0 AND N5=0 THEN 1780
1642 PRINT "HOW MANY PRINTED COPIES OF THE SEARCH RESULTS DO YOU WANT";
1643 M4=0: INPUT M4
1644 IF M4<=0 THEN 1642
1645 PRINT "ROLL PAPER TO THE TOP OF THE PAGE AND PRESS RETURN.";
1646 M$="": INPUT M$
1647 IF N5=0 OR N7=2 THEN 1780
1750 LPRINT
1760 LPRINT FILE$;" FILE LISTED ON ";D$
1770 LPRINT
1771 M3=M3+3
1772 GOSUB 3910
1780 V7=1
1790 IF N7<>1 THEN 1570
1800 IF N4=1 THEN 2010
1810 N4=1
1830 IF N5<>0 THEN 1850
1840 GOTO 2090
1850 M7=M7+1
1851 IF M7<>1 THEN 1890
1855 N3=INT(NN/N5)
1890 N2=0
1900 V2=1
1910 IF V(V2)=0 THEN 1950
1920 IF S$(V2)="" THEN 1940
1930 LPRINT TAB(N3*N2);MID$(S$(V2),1,N3);
1940 N2=N2+1
1950 V2=V2+1
1960 IF N2=N5 THEN 1980
1970 GOTO 1910
1980 LPRINT
1990 LPRINT
1991 M3=M3+2
1992 GOSUB 3910
2010 N2=0
2015 IF N5=0 THEN 2090
2020 FOR S2=1 TO MNUM
2030 IF V(S2)=0 THEN 2070
2040 IF E$(S2)="" THEN 2060
2041 M9=LEN(E$(S2))
2042 FOR M8=1 TO M9
2043 IF MID$(E$(S2),M8,1)<>" " THEN 2050
2044 NEXT M8
2045 GOTO 2060
2050 LPRINT TAB(N3*N2);MID$(E$(S2),1,N3-1);
```

PSEAR CONTINUED

```
2060 N2=N2+1
2070 NEXT S2
2080 LPRINT
2081 M3=M3+1
2082 GOSUB 3910
2090 N6=N6+1
2100 GOTO 1100
2120 IF N5=0 THEN 2170
2122 PRINT "ARE THE RESULTS TO BE LISTED IN COLUMN FORM (YES OR NO)";
2130 Z$="": INPUT Z$
2140 IF LEFT$(Z$,1)="Y" THEN 2170
2150 IF LEFT$(Z$,1)<>"N" THEN 2122
2160 PRINT "DO YOU WANT LABELS (YES OR NO)";: Z$="": INPUT Z$
2161 IF LEFT$(Z$,1)="N" THEN 2180
2162 IF LEFT$(Z$,1)<>"Y" THEN 2160
2163 N7=2: RETURN
2170 N7=1
2180 RETURN
2190 RESUME 2195
2191 IF PSIZE=0 OR M3<PSIZE-5 THEN 2195
2192 M3=PSIZE+1
2193 GOSUB 3910
2195 IF N7=2 THEN 2330
2196 LPRINT
2200 LPRINT
2210 LPRINT N6;" OF THE ";N8;" ITEMS IN THE FILE MEET THE REQUIREMENTS"
2220 LPRINT
2230 LPRINT
2231 M3=M3+5
2240 IF V4=0 THEN 2330
2250 FOR V5=1 TO I1
2260 IF R(V5)<>5 THEN 2280
2270 LPRINT "TOTALS UNDER HEADING ";S$(H(V5));" = ";W(V5)
2271 IF N6<>0 THEN LPRINT "   AVERAGE OF VALUES FOUND = ";W(V5)/N6
2272 IF N6<>0 THEN M3=M3+2 ELSE M3=M3+1
2273 GOSUB 3910
2280 NEXT V5
2290 LPRINT
2320 LPRINT
2325 M3=M3+2
2326 GOSUB 3910
2330 M4=M4-1
2331 FOR I=1 TO SIZE: W(I)=0
2332 NEXT I
2333 CASE=2
2334 GOTO 80
2335 IF PSIZE=0 THEN GOSUB 3972
2339 IF PSIZE=0 THEN V7=0: M3=0: N4=0: N6=0: N8=0
2340 IF M4>=1 THEN 1100
2341 PRINT "DO YOU WANT TO PERFORM ANOTHER SEARCH (YES OR NO)";
```



```
2342 A$="": INPUT A$
2343 IF LEFT$(A$,1)="N" THEN 4010
2344 IF LEFT$(A$,1)<>"Y" THEN 2341
2345 CASE=1: IF PSIZE <>0 THEN M4=1
2346 GOTO 61
2450 FOR V1=1 TO SIZE
2470 W(V1)=0
2480 NEXT V1
2481 FOR V1=1 TO HNUM
2482 V(V1)=0
2483 NEXT V1
2490 PRINT
2500 PRINT
2510 PRINT "NOW SPECIFY THE NUMBERS OF ALL THE HEADINGS YOU WANT"
2520 PRINT "PRINTED  ENTER 0 WHEN DONE, OR -1 FOR ALL THE HEADINGS."
2530 FOR V1=1 TO SIZE
2540 N=-2: INPUT N
2550 IF N=-1 THEN 2660
2560 IF N=0 THEN 2700
2570 IF N<1 OR N>HNUM THEN 2630
2590 IF V(N)=1 THEN 2640
2600 V(N)=1
2610 N5=N5+1
2620 GOTO 2640
2630 PRINT "ENTER ONLY -1, 0, OR A NUMBER FROM 1 TO ";HNUM
2640 NEXT V1
2650 GOTO 2700
2660 FOR V1=1 TO HNUM
2670 V(V1)=1
2680 NEXT V1
2690 N5=HNUM
2700 RETURN
2740 K8=0
2741 FOR V6=1 TO 3
2742 K6(V6)=0
2743 NEXT V6
2744 FOR V6=1 TO LEN(E$(H(I)))
2750 FOR V5=1 TO LR
2760 IF MID$(E$(H(I)),V6,1)=MID$(R$,V5,1) THEN 2790
2770 NEXT V5
2780 GOTO 1320
2790 IF V5=1 THEN 2795
2791 IF V5<12 THEN 2794
2792 IF K6(15-V5)=1 THEN 1320 ELSE K6(15-V5)=1
2793 GOTO 2795
2794 K8=2
2795 NEXT V6
2800 IF E$(H(I))="" THEN 1320
2805 IF K8=0 THEN 1320
2810 W(I)=W(I)+VAL(E$(H(I)))
```

PSEAR CONTINUED

```
2820 GOTO 1360
2850 PRINT "YOU WILL BE ASKED TO SPECIFY THE HEADING NUMBERS OF ALL THE"
2855 PRINT "HEADINGS YOU WANT TO SEARCH UNDER, AND THE PARTICULAR TYPE"
2860 PRINT "OF SEARCH YOU WANT TO CONDUCT.  THE SEARCHES CAN BE OF"
2865 PRINT "SIX TYPES.  1) LOOK FOR ALL EXACT MATCHES TO A SPECIFIED"
2870 PRINT "STANDARD UNDER THAT HEADING, 2) LOOK FOR ALL ITEMS UNDER"
2875 PRINT "THE HEADING THAT HAVE THE SAME FIRST LETTERS AS THE"
2880 PRINT "SPECIFIED STANDARD, 3) LOOK FOR ALL ITEMS UNDER THE"
2885 PRINT "HEADING THAT HAVE A VALUE LESS THAN OR EQUAL TO A SPECIFIED"
2890 PRINT "STANDARD, 4) LOOK FOR ALL ITEMS UNDER THE HEADING THAT HAVE"
2895 PRINT "A VALUE GREATER THAN OR EQUAL TO A SPECIFIED STANDARD, "
2900 PRINT "5) LOOK FOR ALL ITEMS UNDER THE HEADING THAT HAVE A"
2905 PRINT "NUMERICAL EQUIVALENT; BE CAREFUL, IF AN ENTRY CONTAINS"
2910 PRINT "2 NUMBERS WITH BLANKS IN BETWEEN, THEY WILL BE TREATED AS"
2915 PRINT "ONE NUMBER--E.G. '12 34' WOULD BE INTERPRETED AS '1234'. "
2920 PRINT "AS A SIDE BONUS, YOU GET THE TOTAL FOR ALL OF THE NUMBERS"
2925 PRINT "FOUND UNDER THE HEADING THAT MEET THE REST OF THE SEARCH"
2930 PRINT "CRITERIA (IF ANY).  6) LOOK FOR A PARTICULAR STANDARD"
2935 PRINT "ANYWHERE WITHIN THE FIELD--E.G.  IF THE DATA WERE 'ABEL'"
2940 PRINT "'BEETOVEN' 'BACH' 'JEZABEL' A STANDARD OF 'BE' WOULD"
2945 PRINT "FIND 'ABEL' 'BEETOVEN' AND 'JEZABEL' BUT NOT 'BACH'. "
2946 PRINT
2947 PRINT"PRESS RETURN FOR MORE HELP, ANYTHING ELSE TO GO BACK. ";
2948 Y$="": INPUT Y$
2949 IF Y$<>"" THEN RETURN
2950 PRINT "  YOU MAY ASK FOR A HEADING TO BE SEARCHED IN MORE"
2955 PRINT "THAN ONE WAY.  HOWEVER, A MAXIMUM OF ";SIZE;" SEARCH"
2960 PRINT "HEADINGS HAS BEEN IMPOSED.  ONCE YOU HAVE SPECIFIED"
2965 PRINT "ALL OF THE HEADINGS YOU WANT SEARCHED, THEIR CORRESPON-"
2970 PRINT "DING SEARCH TYPES, AND THEIR RELATED STANDARDS (IF"
2975 PRINT "NECESSARY), YOU MUST ALSO INDICATE 1) IF ALL OF THE"
2980 PRINT "CONDITIONS SPECIFIED MUST BE MET BEFORE AN ITEM IS"
2985 PRINT "INCLUDED (I.E. AN 'AND' SEARCH), OR IF 2) MEETING ANY"
2990 PRINT "OF THE CRITERIA IS ENOUGH TO INCLUDE AN ITEM (I.E. AN"
2995 PRINT "'OR' SEARCH TYPE) (UNLESS YOU ARE ONLY SEARCHING UNDER"
3000 PRINT "ONE HEADING, IN WHICH CASE IT DOESN'T MATTER).  ALSO, "
3005 PRINT "YOU ARE GIVEN THE CHOICE AS TO WHICH HEADINGS TO HAVE"
3010 PRINT "DISPLAYED FOR THE ITEMS THAT MATCH THE SEARCH CRITERIA"
3015 PRINT "(I.E. THE PRINTED HEADINGS NEED NOT BE THE SAME AS THE"
3020 PRINT "SPECIFIED HEADINGS IN THE SEARCH). "
3021 PRINT
3022 PRINT "PRESS RETURN FOR MORE HELP, ANYTHING ELSE TO GO BACK. ";
3023 Y$="": INPUT Y$
3024 IF Y$<>"" THEN RETURN
3025 PRINT "THE PROGRAM WILL ALSO ASK YOU IF"
3026 PRINT "YOU WANT THE RESULTS LISTED IN A TABULAR"
3030 PRINT "(COLUMN) FORM, AS OPPOSED TO A ROW (LINE BY LINE) FORM. "
3035 PRINT "YOU WILL BE ASKED WHETHER OR NOT YOU ARE USING A PAPER-"
3040 PRINT "PRINTING TERMINAL; IF YOU ARE, NO PAGING WILL BE DONE. "
3045 PRINT "IF YOU ARE USING A TV-LIKE-TERMINAL, PAGING WILL OCCUR"
3050 PRINT "(I.E. ONLY ONE SCREEN-FULL OF INFORMATION WILL BE SENT"
3055 PRINT "TO YOU AT A TIME; YOU CAN CHOOSE IF YOU WANT MORE SENT). "
3060 PRINT "YOU HAVE THE OPTION OF SEARCHING THROUGH THE FILE AS"
3065 PRINT "MANY TIMES AS YOU WANT TO, AND OF GETTING AS MANY"
3070 PRINT "COPIES OF THE PRINTED RESULTS AS YOU LIKE. "
3075 PRINT
3080 RETURN
3170 GOTO 4010
3210 PRINT "WHAT IS THE NAME OF THE DATA FILE IN DRIVE 1"
3230 FILE$="": INPUT FILE$
3240 IF FILE$="" THEN 3210
3250 FILE$=FILE$+": 1"
3260 RETURN
3910 IF PSIZE=0 OR M3<PSIZE THEN RETURN
3920 M3=0
3930 PRINT "PRESS RETURN TO CONTINUE, ANYTHING ELSE TO STOP";
3940 Y$="": INPUT Y$
3950 IF Y$="" THEN RETURN
3960 GOTO 2330
3972 IF M5>=M3 THEN 3980
3974 M3=M3-M5
3976 GOTO 3972
3980 IF M5=M3 OR N7=2 THEN RETURN
3981 FOR SKIP=1 TO M5-M3
3985 LPRINT
3987 NEXT SKIP
3989 RETURN
3991 DATA 50
3992 DATA 80
3993 DATA 132
3994 DATA 66
3995 DATA 22
3996 DATA "DATA"
4010 CLOSR
4011 ON ERROR GOTO 4014
4012 OPENR "EXEC"
4013 CLOSR: GOTO 4020
4014 RESUME 4015
4015 CLOSR
4016 PRINT "PLACE A DISKETTE LABELLED 'PROGRAMS' IN DRIVE 0. "
4017 PRINT "THEN PRESS THE RETURN KEY ON THE TERMINAL. "
4018 A$="": INPUT A$
4019 GOTO 4011
4020 END
```

60

WORLD

If memory size is limited, the maximum number of search criteria can be changed to save space by altering lines 17 and 20. If another type of search (for example: less than, greater than, equal to, etc.) is desired, instead of the present version which scans data items for a specified standard, lines 390-400 could be changed to permit a different search algorithm.

The following list contains all the variables used in WORLD and the function each variable serves.

Name	Descrition
A	inputs heading numbers for averaging
B$	used to check for return (when user no longer wants more data)
C(L4)	computes totals for all possible combinations
C1(L4,L6)	computes totals for all combinations of headings searched
C2(L4,L6)	computes the number of all entries for each total in C1
D(L)	stores temporary flags
G(L)	stores heading numbers to be searched
H	number of headings in the data file
H$(H)	stores heading names in data file
I	loop parameter
J	loop parameter
L	maximum number of headings which can be searched
L1	maximum number of standards which can be declared for each heading
L2	counts the number of headings requested
L4	computes the total number of combinations in the matrix to be found
L5	index into the array, C
L6	maximum number of headings for which averages can be computed
L7	counts the number of headings for which averages are requested
M	loop parameter
P	temporary product used for indexing C
PSIZE	number of lines per page on the system terminal
S$(L,L1)	stores standards for each search criterion
T	counts the number of standards requested for each heading; later used to count how many lines have been output
X(L)	number of standards for each heading
Y(L6)	heading numbers for which averages are requested

WORLD LISTING

```
1 REM *** PROGRAMMED BY VIC MOIZNIEKS 5/78
2 REM *** SEARCH DATA FILE IN 'CROSS-TAB' FASHION
3 REM *** OPTIONALLY COMPUTE HEADING TOTALS AND AVERAGES
4 REM ***
5 CLEAR 2000  'RESERVE STRING-HANDLING AREA
10 PRINT "ALWAYS PRESS RETURN AFTER TYPING A RESPONSE!":PRINT:PRINT
15 PSIZE=20  'NUMBER OF LINES TO BE PRINTED PER PAGE
17 L=10  'MAXIMUM NUMBER OF HEADINGS THAT CAN BE SPECIFIED
20 L1=20  'MAXIMUM NUMBER OF STANDARDS POSSIBLE PER HEADING
25 L6=5  'MAXIMUM NUMBER OF HEADINGS FOR WHICH TO RECEIVE AVERAGES
30 PRINT "NAME OF DATA FILE (PROBABLY GRAD1 OR GRAD2)";
40 F$="":INPUT F$
50 IF F$="" THEN 30
55 REM *** TAKE PAINS TO ENSURE USER TYPES IN LEGAL FILE NAME
60 ON ERROR GOTO 75
70 OPENR F$+":1":GOTO 90
75 RESUME 80
80 PRINT F$;" DOES NOT EXIST ON DRIVE 1  TRY AGAIN":CLOSR:GOTO 30
90 ON ERROR GOTO 330
93 REM *** Y = HEADING NUMBERS FOR TOTALS/AVERAGES
94 REM *** X = NUMBER OF STANDARDS SPECIFIED PER HEADING
95 REM *** H = # OF HEADINGS IN DATA FILE
96 REM *** H$ = HEADING NAMES (LATER USED TO HOLD DATA ITEM)
97 REM *** S$ = STANDARDS SPECIFIED FOR EACH HEADING
98 REM *** D = TEMPORARY FLAG VARIABLES
99 REM *** G = HEADING NUMBRS ASKED TO BE SEARCHED
100 DSKIN H:DIMH$(H),S$(L,L1),D(L),G(L),X(L),Y(L6):X(0)=0
105 REM *** READ DATA FILE HEADING INFORMATION
110 FOR I=1 TO H:H$(I)="":DSKIN H$(I):NEXT I
120 PRINT "TYPE HEADING NUMBER YOU WANT INCLUDED, OR 0 FOR LIST OF"
125 PRINT "HEADINGS, OR -1 TO START ALL OVER, OR";H+1;"TO STOP"
130 PRINT "SPECIFYING HEADINGS  YOU CAN SPECIFY A MAXIMUM OF";L
135 PRINT "HEADING NUMBERS ":L2=0:L4=1
139 REM *** GET HEADING NAMES AND STANDARDS
140 FOR I=1 TO L
150 A=-2:INPUT A
155 REM *** CHECK HEADING VALUE ENTERED BY USER
160 IF A<-1 OR A>H+1 THEN PRINT "ERROR":GOTO 150  'ILLEGAL?
170 IF A=-1 THEN 120  'START ALL OVER?
175 IF A=H+1 THEN 815  'ALL DONE SPECIFYING HEADINGS?
180 IF A<>0 THEN 270  'LIST OF HEADING NAMES AND NUMBERS?
185 T=0  'COUNTER FOR NUMBER OF LINES PRINTED ON TERMINAL
190 FOR J=1 TO H  'PRINT ALL HEADING NAMES
200 IF T<=PSIZE THEN 240  'CHECK FOR FULL PAGE ON TERMINAL
210 PRINT "PRESS RETURN FOR MORE, ANYTHING ELSE TO STOP";B$="":INPUT B$
220 IF B$="" THEN T=0:GOTO 240
230 IF A=0 THEN 150 ELSE RETURN  'SEE IF PRINTING FOR TOTALS
240 PRINT J;H$(J):T=T+1  'PRINT HEADING NUMBER AND VALUE ON TERMINAL
250 NEXT J
260 IF A=0 THEN 150 ELSE RETURN 'SEE IF PRINTING FOR TOTALS
270 L2=L2+1:G(I)=A 'COUNT # OF HEADINGS ASKED FOR, STORE REAL HEADING #
271 PRINT "TYPE STANDARD(S) TO BE LOOKED FOR UNDER HEADING";A;"(";H$(A);
272 PRINT")."
273 PRINT "IF STANDARD PRESENT ANYWHERE IN FIELD, ITEM WILL BE COUNTED."
274 PRINT "TYPE DONE WHEN FINISHED, OR OVER TO START OVER.":T=0
275 FOR J=1 TO L1  'ALLOW USER TO SPECIFY MAXIMUM NUMBER OF STANDARDS
280 S$(I,J)="":INPUT S$(I,J)
290 IF S$(I,J)="OVER" THEN 271  'WANT TO START OVER?
300 IF S$(I,J)="DONE" AND J<>1 THEN 320  'ALL DONE SPECIFYING STANDARDS?
301 IF S$(I,J)="DONE" THEN PRINT "YOU MUST SPECIFY AT LEAST ONE VALUE"
    :GOTO 280  'NEED AT LEAST ONE STANDARD PER HEADING
305 T=T+1  'INCREASE COUNTER ON NUMBER OF STANDARDS SPECIFIED
310 NEXT J  'GET NEXT STANDARD
315 REM *** COUNT TOTAL NUMBER OF MATRIX VALUES TO BE FOUND (L4)
316 REM *** STORE NUMBER OF STANDARDS SPECIFIED PER HEADING (X)
320 L4=L4*T:X(I)=J-1:PRINT "NEXT HEADING";:NEXT I
321 GOTO 815  'USER SPECIFIED MAXIMUM NUMBER OF ALLOWED HEADINGS
325 REM *** ERROR SUBROUTINE USED IF FILE DOES NOT HAVE PROPER FORMAT
330 RESUME 335
335 PRINT "ERROR---FILE ";F$;" NOT IN PROPER FORMAT FOR THIS PROGRAM."
340 CLOSR: STOP
344 REM *** NON-END OF FILE ERROR CAUSING PROGRAM TERMINATION
345 PRINT "AN UNEXPLAINABLE ERROR OCCURRED IN LINE";ERL;" ";ERR
346 PRINT "PLEASE EXPLAIN WHAT HAPPENED TO THE PERSON IN CHARGE."
347 CLOSR:STOP
348 REM *** C HOLDS TOTALS FOR ALL STANDARD COMBINATIONS
349 REM *** C1 HOLDS TOTALS FOR ALL COMBINATION HEADINGS ASKED FOR
350 DIM C(L4),C1(L4,L6),C2(L4,L6)
351 REM *** C2 HOLDS NUMBER OF ENTRIES FOR ALL TOTALS IN C1
352 REM *** READ DATA FILE UNTIL ENCOUNTER EOF ERROR (27)
355 ON ERROR GOTO 500  'LOOK FOR END OF FILE
359 REM *** READ DATA ITEM INTO H$ ARRAY FOR COMPARISON(S)
360 FOR I=1 TO H:H$(I)="":DSKIN H$(I):NEXT I
365 REM *** COMPARE ALL HEADINGS SPECIFIED (L2)
370 FOR J=1 TO L2
375 REM *** LOOK FOR ALL POSSIBLE STANDARDS UNDER EACH HEADING
380 FOR I=1 TO L1
385 IF S$(J,I)="DONE" THEN 355  'DATA ITEM NOT OF TYPE REQUESTED?
386 IF S$(J,I)<>"" THEN 390  'NULL STRING SPECIAL CASE
387 IF H$(G(J))="" THEN D(J)=I:GOTO420  'EXACTLY EQUAL TO NULL STRING?
388 GOTO 410 'DATA ITEM DOES NOT MEET NULL STRING REQUIREMENT
390 D(J)=INSTR(H$(G(J)),S$(J,I))  'STANDARD PRESENT IN DATA FIELD?
400 IF D(J)<>0 THEN D(J)=I: GOTO 420  'IF SO, RECORD VALUE
410 NEXT I  'KEEP LOOKING THROUGH SPECIFIED STANDARDS
415 GOTO 355  'STANDARD NOT FOUND, SO REJECT DATA ITEM
420 NEXT J  'LOOK AT NEXT SPECIFIED HEADING
421 L5=0  'INDEX INTO C ARRAY, INITIALLY SET TO 0
422 FOR I=1 TO L2  'FOR ALL SPECIFIED HEADINGS
```

WORLD CONTINUED

```
423 IF I=1 THEN 429  'CHECK IF LOOKING AT FIRST HEADING
424 P=1  'TEMPORARY PRODUCT FOR INDEXING INTO C ARRAY
425 FOR M=1 TO I-1  'ADD UP ALL ELEMENTS BEFORE CURRENT ONE FOR INDEX
426 P=P*X(M)  'BY TAKING ALL NECESSARY PRODUCTS
427 NEXT M
428 L5=L5+P*(D(I)-1):GOTO 430  'UPDATE INDEX VALUE
429 L5=L5+D(I)  'SPECIAL CASE OF INDEX VALUE (1ST HEADING)
430 NEXT I
431 IF L7=0 THEN 440  'ASKED FOR HEADING AVERAGES?
432 FOR I=1 TO L7  'IF SO, COMPUTE TOTALS FOR ALL HEADINGS ASKED FOR
433 T=VAL(H$(Y(I)))  'SEE IF NUMBER (NOT EQUAL TO 0) PRESENT
434 IF T=0 THEN 436  'IF VALUE = 0, REJECT FROM TOTAL
435 C1(L5,I)=C1(L5,I)+T: C2(L5,I)=C2(L5,I)+1  'UPDATE TOTAL & COUNTERS
436 NEXT I  'GET NEXT HEADING NUMBER
440 C(L5)=C(L5)+1  'INCREASE COUNTER IN MATRIX USING INDEX VALUE
450 GOTO 355  'GET NEXT DATA ITEM FROM FILE
500 PRINT:PRINT:IF ERR=27 THEN RESUME 510 ELSE 345 'MAKE SURE EOF ERROR
505 REM *** SEARCH COMPLETE, TIME TO PRINT OUT MATRIX
506 REM *** T COUNTS NUMBER OF LINES PRINTED ON TERMINAL SCREEN
510 T=0:PRINT "TOTALS (IN HEADING SPECIFIED ORDER)":PRINT:PRINT
540 REM *** INITIALLY START WITH FIRST STANDARD OF EACH HEADING
550 FOR I=1 TO L2: D(I)=1: NEXT I
590 REM *** INDEX THROUGH ALL COMBINATIONS OF STANDARDS
600 FOR M=1 TO L4  'FOR ALL COMBINATIONS
610 FOR I=1 TO L2 'PRINT ONE STANDARD FROM EACH HEADING SET OF STANDARDS
620 PRINT S$(I,D(I));"  ";
630 NEXT I
640 PRINT "=";C(M):IF L7>0 THEN GOSUB 840 'NEED TO PRINT ANY AVERAGES?
641 T=T+1:IF T<=PSIZE THEN 650  'CHECK ON FULL TERMINAL PAGE
642 PRINT "PRESS RETURN FOR MORE, ANYTHING ELSE TO STOP";
643 B$="":INPUT B$
644 IF B$<>"" THEN 990
645 T=0  'RESET LINES PRINTED ON TERMINAL COUNTER FOR NEXT PAGE
650 FOR I=1 TO L2  'UPDATE WHICH COMBINATION OF STANDARDS TO USE
660 D(I)=D(I)+1  'BY PRINTING IN PREDETERMINED ORDER
670 IF D(I)<=X(I) THEN 800  'CHECK TO SEE IF EXCEEDED NUMBER SPECIFIED
680 D(I)=1 'IF SO, RESET INDEX AND 'LOOK AGAIN' FOR NEXT INDEX TO UPDATE
690 NEXT I
800 NEXT M  'PRINT NEXT COMBINATION
810 GOTO 990  'AND STOP
811 REM *** GET HEADING NUMBERS FOR WHICH TOTALS/AVERAGES ARE TO BE
812 REM *** COMPUTED FOR ALL COMBINATIONS OF SPECIFIED STANDARDS
813 REM *** L7 COUNTS NUMBER OF HEADINGS ASKED FOR
815 PRINT "TYPE HEADING NUMBER(S) YOU WANT TOTAL/AVERAGE FOR, OR TYPE"
816 PRINT"0 FOR A LIST OF HEADING NUMBERS, OR TYPE -1 TO START OVER, OR"
817 PRINT "TYPE";H+1;"TO STOP SPECIFYING HEADING NUMBERS.":L7=0
820 FOR I=1 TO L6  'ASK FOR UP TO MAXIMUM ALLOWED NUMBER OF HEADINGS
825 A=-2:INPUT A
830 IF A<-1 OR A>H+1 THEN PRINT "ERROR ":GOTO 455  'ILLEGAL HEADING?
835 IF A<>0 THEN 840  'ASKED FOR LIST OF HEADING NUMBERS?
836 A=-2:GOSUB 185  'GO PRINT LIST OF HEADING NUMBERS
837 GOTO 825  'AND GET NEXT HEADING NUMBER USER WANTS
840 IF A=-1 THEN 813  'START OVER
843 IF A=H+1 THEN 350  'ALL DONE SPECIFYING HEADING NUMBERS?
845 L7=L7+1:Y(I)=A  'UPDATE COUNTER ON NUMBER OF HEADINGS SPECIFIED
846 REM *** AND STORE REAL HEADING NUMBER ASKED FOR IN Y ARRAY
850 NEXT I  'GET NEXT HEADING FROM USER
855 GOTO 350  'USER SPECIFIED MAXIMUM NUMBER OF ALLOWED HEADINGS
859 REM *** PRINT OUT TOTALS AND AVERAGE FOR ALL HEADINGS ASKED FOR
860 FOR I=1 TO L7  'FOR ALL HEADINGS USER SPECIFIED
865 PRINT"   TOTAL UNDER HEADING";Y(I);"=";C1(M,I);"(N=";C2(M,I);").  ";
870 IF C2(M,I)=0 THEN C2(M,I)=1  'GUARD AGAINST ZERO-DIVIDE
875 PRINT "AVERAGE VALUE =";C1(M,I)/C2(M,I)
890 T=T+1:IF T<=PSIZE THEN 940  'CHECK FOR FULL TERMINAL SCREEN
900 PRINT "PRESS RETURN FOR MORE, ANYTHING ELSE TO STOP";
910 B$="":INPUT B$
920 IF B$<>"" THEN 990  'USER SEEN ALL HE/SHE WANTS TO?
930 T=0  'IF NOT, RESET NUMBER OF LINES PRINTED ON TERMINAL COUNTER
940 NEXT I  'AND CONTINUE PRINTING
950 RETURN
980 REM *** FINISH UP BY CLOSING FILE
990 CLOSE  'CLOSE DATA FILE SO PROGRAM CAN BE RUN AGAIN
1000 END
```

PULL

If different search criteria are desired, lines 1020, 1040, 1580, and 1600 can be changed to fit the type of search you want. The maximum number of search constraints can be changed by altering line 2440.

The following list contains all the variables used in PULL and the function each variable serves.

Name	Description
A$	inputs answers to system messages
F$	name of the data file
G$	name of the new file for output
H	number of headings in the data file
H$(H)	stores heading names; later used to store all data on the entry currently being processed
I	loop parameter
I$	number of the drive in which the data file is located
J	loop parameter
J$	number of the drive in which the new file is located
K	used to input heading numbers to be searched
M	maximum number of search criteria allowed
S	used to count the number of headings to be placed in the new file
T(H)	flag to indicate which data headings go into the new file
X(M)	stores search types
Y$(M)	stores standard for each search criterion
Z(M)	used to specify search constraints

FULL LISTING

```
10 REM *** PROGRAMMED BY VIK MUIZNIEKS 1/78
20 REM *** THIS PROGRAM ALLOWS SUBSETS OF DATA FILES TO BE CREATED FROM
30 REM *** DATA FILES ALREADY CREATED VIA THE INFO PACKAGE.  ORIGINAL
40 REM *** FILES CAN ALSO BE ALTERED TO REMOVE ITEMS OF THE SAME SUBSET.
50 CLEAR
60 CLEAR 2000
70 REM *** GET FILE NAMES; BE CAREFUL ABOUT USER TYPING ERRORS
80 PRINT "NAME OF FILE CONTAINING DATA ";
90 F$="":INPUT F$
100 IF F$="" THEN 60
110 PRINT "IN WHICH DRIVE IS THE DISKETTE CONTAINING THE DATA (0 OR 1)";
120 I$="": INPUT I$
130 IF I$="0" THEN 150
140 IF I$<>"1" THEN 110
150 ON ERROR GOTO 180
160 OPENR F$+":"+I$
170 GOTO 220
180 RESUME 190
190 PRINT"ERROR-FILE ";F$;" DOES NOT EXIST ON THE DISKETTE IN DRIVE ";I$
200 CLOSR
210 GOTO 90
220 PRINT "WHAT IS THE NAME OF THE FILE YOU ARE ABOUT TO CREATE";
230 G$="":INPUT G$
240 IF G$="" THEN 220
250 PRINT"WHICH DRIVE HAS THE DISKETTE CONTAINING FILE ";G$;" (0 OR 1)";
260 J$="": INPUT J$
270 IF J$="0" THEN 290
280 IF J$<>"1" THEN 250
290 ON ERROR GOTO 320
300 OPENW G$+":"+J$
310 GOTO 370
320 RESUME 330
330 PRINT "ERROR-SPACE HAS NOT BEEN ALLOCATED FOR FILE ";G$;" ON THE"
340 PRINT "DISKETTE IN DRIVE ";J$;".  SEE THE FDOS ALLOC COMMAND."
350 CLOSW
360 GOTO 220
370 ON ERROR GOTO 390
380 GOTO 460
390 PRINT "ERROR #";ERR;"IN LINE #";ERL
400 CLOSR
410 CLOSW
420 PRINT "I DON'T KNOW WHAT HAPPENED, BUT SOMETHING WENT WRONG."
430 PRINT "PLEASE WRITE DOWN THE ABOVE INFORMATION AND TURN IT IN"
440 STOP
450 REM *** GET DATA FILE INFORMATION
460 M=0: DSKIN M
470 REM *** M HAS # OF HEADINGS IN FILE
480 REM *** H$ HAS HEADING NAMES, USED LATER FOR DATA ELEMENTS IN FILE
490 REM *** T HOLDS FLAGS FOR WHICH DATA FIELDS TO PRINT IN NEW FILE
500 REM *** M CONTAINS # OF FILE CONSTRAINTS USER IS ALLOWED TO SPECIFY
510 DIM H$(M),T(M)
520 READ M
530 DIM X(M),Y$(M),Z(M)
540 REM *** X,Y$,Z USED TO SPECIFY FILE CONSTRAINTS
550 FOR I=1 TO M
560 H$(I)="":DSKIN H$(I)
570 NEXT I
580 REM *** FIND OUT WHAT FIELDS USER WANTS PUT IN NEW FILE.
590 GOSUB 1730
600 PRINT "DO YOU KNOW HOW TO SET UP FILE CONSTRAINTS (Y OR N)";
610 A$="": INPUT A$
620 IF LEFT$(A$,1)="Y" THEN 760
630 IF LEFT$(A$,1)<>"N" THEN 600
640 PRINT "TYPE IN THE HEADING NUMBER, STANDARD, OPTION (0 OR 1)"
650 PRINT "FOR EACH STANDARD YOU WANT TO SPECIFY UNDER EACH HEADING."
660 PRINT "IF THE OPTION IS 1, THE STANDARD MUST BE PRESENT (SOMEWHERE)"
670 PRINT "IN THE FIELD UNDER THE SPECIFIED HEADING NUMBER IN ORDER FOR"
680 PRINT "THE DATA TO BE PLACED IN FILE ";G$;".  IF THE OPTION"
690 PRINT "IS 0, THE STANDARD CANNOT BE PRESENT ANYWHERE WITHIN THE"
700 PRINT "FIELD UNDER THE SPECIFIED HEADING IN ORDER FOR THE "
710 PRINT "DATA TO BE PLACED IN FILE ";G$;"."
720 PRINT " E.G.  2,SEED,1 WOULD ASK FOR ALL ENTRIES THAT HAVE SEED IN"
730 PRINT "                THE SECOND FIELD TO BE PLACED IN FILE ";G$
740 PRINT "       3,C7,0   WOULD ASK THAT ENTRIES HAVING A C7 SOMEWHERE"
750 PRINT "                IN THE THIRD FIELD BE EXCLUDED FROM FILE ";G$
760 PRINT "TYPE IN THE HEADING NUMBER, STANDARD, OPTION (0 OR 1)"
770 PRINT"TYPE IN 0,0,0 WHEN DONE (NOTE:  YOU CAN SPECIFY A MAXIMUM OF";M
780 PRINT "CONSTRAINTS."
790 FOR I=1 TO M
800 GOTO 820
810 PRINT "ERROR, PLEASE RETYPE."
820 X(I)=-1:Y$(I)="":Z(I)=-1:INPUT X(I),Y$(I),Z(I)
830 IF X(I)=0 AND Y$(I)="0" AND Z(I)=0 THEN 890
840 IF X(I)<1 OR X(I)>M THEN 810
850 IF Z(I)<0 OR Z(I)>1 THEN 810
860 IF Y$(I)="" THEN 810
870 NEXT I
880 I=M+1
890 IF I>1 THEN 930
900 PRINT "YOU MUST SPECIFY AT LEAST ONE SEARCH CONSTRAINT!!"
910 GOTO 790
920 REM *** SCAN FILE AND MATCH AGAINST USER CONSTRAINTS.
930 ON ERROR GOTO 1150
940 REM *** READ IN ONE ENTRY
950 FOR J=1 TO M
960 H$(J)="":DSKIN H$(J)
```

PULL CONTINUED

```
970 NEXT J
980 REM *** SEE IF ALL FIELDS FIT CONSTRAINTS
990 FOR J=1 TO I-1
1000 IF Z(J)=0 THEN 1040
1010 REM *** INSTR IS 'LOOK FOR SUBSTRING OF A STRING' FUNCTION
1020 IF INSTR(H$(X(J)),Y$(J))<>0 THEN 1060
1030 GOTO 930
1040 IF INSTR(H$(X(J)),Y$(J))=0 THEN 1060
1050 GOTO 930
1060 NEXT J
1070 REM *** ALL CONSTRAINTS MET, PLACE APPROPRIATE ENTRY IN FILE
1080 FOR J=1 TO H
1090 IF T(J)=0 THEN 1110
1100 DSKOUT H$(J)
1110 NEXT J
1120 REM *** GO LOOK AT NEXT ENTRY
1130 GOTO 930
1140 REM *** FILE SHOULD BE ALL READ, CLOSE UP ALL FILES
1150 IF ERR=27 THEN RESUME 1160 ELSE 390
1160 CLOSR
1170 CLOSW
1180 PRINT "DO YOU WANT TO REMOVE THE DATA ITEMS YOU JUST COPIED FROM"
1190 PRINT "FILE ";F$;" (Y OR N)";
1200 A$="": INPUT A$
1210 IF LEFT$(A$,1)="N" THEN 2110
1220 IF LEFT$(A$,1)<>"Y" THEN 1180
1230 PRINT"DRIVE";1-VAL(I$);"SHOULD HAVE A DISKETTE LABELLED 'SCRATCH'"
1240 PRINT "WHEN READY, PRESS G AND THEN RETURN.  IF YOU CHANGED YOUR"
1250 PRINT "MIND, JUST PRESS RETURN "
1260 A$="": INPUT A$
1270 IF A$="" THEN 2110
1280 IF A$<>"G" THEN 1230
1290 REM *** TAKE PAINS TO MAKE SURE ALL FILES IN RIGHT PLACES
1300 ON ERROR GOTO 1380
1310 OPENW "SCR:"+RIGHT$(STR$(1-VAL(I$)),1)
1320 ON ERROR GOTO 1350
1330 OPENR F$+":"+I$
1340 GOTO 1450
1350 RESUME 1360
1360 CLOSR
1370 GOTO 1390
1380 RESUME 1390
1390 PRINT"THE FILES (DISKETTES) ARE NOT ON THE RIGHT DRIVES.  FILE ";F$
1400 PRINT "SHOULD BE ON THE DISKETTE IN DRIVE ";I$;" AND"
1410 CLOSW
1420 GOTO 1230
1430 REM*** SCAN FILE AGAIN, THIS TIME KEEP ITEMS THAT WEREN'T KEPT LAST
1440 REM ***TIME.  KEEP ORIGINAL # OF HEADINGS-NOT USER SPECIFIED NUMBER
1450 ON ERROR GOTO 390
1460 DSKIN H
1470 DSKOUT H
1480 FOR J=1 TO H
1490 DSKIN A$
1500 DSKOUT A$
1510 NEXT J
1520 ON ERROR GOTO 1680
1530 FOR J=1 TO H
1540 H$(J)="": DSKIN H$(J)
1550 NEXT J
1560 FOR J=1 TO I-1
1570 IF Z(J)=0 THEN 1600
1580 IF INSTR(H$(X(J)),Y$(J))=0 THEN 1620
1590 GOTO 1520
1600 IF INSTR(H$(X(J)),Y$(J))<>0 THEN 1620
1610 GOTO 1520
1620 NEXT J
1630 FOR J=1 TO H
1640 DSKOUT H$(J)
1650 NEXT J
1660 GOTO 1520
1670 REM *** FILE SHOULD BE EXHAUSTED, RECOPY FROM DUMMY FILE
1680 IF ERR=27 THEN RESUME 1690 ELSE 390
1690 CLOSR
1700 CLOSW
1710 GOTO 2200
1720 REM *** SUB TO SEE WHICH FIELDS USER WANTS IN NEW FILE
1730 PRINT "TYPE IN THE HEADING NUMBERS FOR THE FIELDS YOU WANT"
1740 PRINT "STORED IN FILE ";G$;".  TYPE 0 FOR A LIST OF HEADINGS AND"
1750 PRINT "HEADING NUMBERS, OR TYPE";H+1;"TO STOP SPECIFYING HEADING";
1760 PRINT " NUMBERS."
1770 K=-2: INPUT K
1780 IF K=H+1 THEN 1980
1790 IF K=-1 THEN 1860
1800 IF K=0 THEN 1930
1810 IF K<1 OR K>H THEN 1730
1820 REM *** SET FLAG FOR PRINTING SPECIFIED FIELD
1830 T(K)=1
1840 GOTO 1770
1850 REM *** OUTPUT ALL HEADING INFO INTO NEW FILE; SET ALL FLAGS
1860 DSKOUT H
1870 FOR K=1 TO H
1880 T(K)=1
1890 DSKOUT H$(K)
1900 NEXT K
1910 RETURN
1920 REM *** PRINT LIST OF HEADING NAMES AND NUMBERS
1930 FOR K=1 TO H
1940 PRINT K;"  ";H$(K)
1950 NEXT K
1960 GOTO 1730
1970 REM *** OUTPUT HEADING INFO FOR NEW FILE; SET NEC. FLAGS
1980 S=0
```

PULL CONTINUED

```
1990 FOR K=1 TO H
2000 IF T(K)=1 THEN S=S+1
2010 NEXT K
2020 IF S<>0 THEN 2050
2030 PRINT "YOU MUST STORE AT LEAST ONE FIELD!!"
2040 GOTO 1730
2050 DSKOUT S
2060 FOR K=1 TO H
2070 IF T(K)=1 THEN DSKOUT H$(K)
2080 NEXT K
2090 RETURN
2100 REM *** SEE IF WANT TO CREATE ANOTHER FILE
2110 PRINT "DO YOU WANT TO RUN THIS PROGRAM AGAIN (Y OR N)";
2120 A$="": INPUT A$
2130 IF LEFT$(A$,1)="N" THEN 2330
2140 IF LEFT$(A$,1)<>"Y" THEN 2110
2150 REM *** AVOID RESTARTING PROBLEMS
2160 ERASE H$,T,X,Y$,Z
2170 RESTORE
2180 GOTO 50
2190 REM *** COPY ORIGINAL FILE BACK FROM DUMMY FILE
2200 OPENR "SCR:"+RIGHT$(STR$(1-VAL(I$)),1)
2210 OPENW F$+":"+I$
2220 DSKIN H
2230 DSKOUT H
2240 ON ERROR GOTO 2280
2250 A$="": DSKIN A$
2260 DSKOUT A$
2270 GOTO 2240
2280 IF ERR=27 THEN RESUME 2290 ELSE 390
2290 CLOSR
2300 CLOSW
2310 GOTO 2110
2320 REM *** MAKE SURE FDOS DISKETTE IN DRIVE 0
2330 ON ERROR GOTO 2370
2340 OPENR "EXEC"
2350 CLOSR
2360 GOTO 2450
2370 RESUME 2380
2380 CLOSR
2390 PRINT "PLACE THE DISKETTE LABELLED 'PROGRAMS' IN DRIVE 0."
2400 PRINT "THEN PRESS THE RETURN KEY ON THE TERMINAL."
2410 INPUT A$
2420 GOTO 2330
2430 REM *** M VALUE (# OF FILE CONSTRAINTS USER ALLOWED TO SPECIFY)
2440 DATA 20
2450 END
```

SPLIT

The option of copying the entire file (line numbers 320-400) can be left out of this program since EXPAN serves that purpose.

The following list contains all the variables used in SPLIT and the function each variable serves.

Name	Description
A$	used to input data from the old file; later used to store the standard, if needed
B$	used to input data from the old file
C	used to input responses to system messages for routing
FILE$	name of the old data file
H	stores number of headings in old file
I	loop parameter
I$	used to input the drive number for data files
N$	name of the new data file

SPLIT LIST!!6

```
1 REM *** PROGRAMMED BY VIK MUIZNIEKS 1/78
2 REM *** THIS PROGRAM ALLOWS USER TO BREAK BIG FILE UP INTO TWO SMALLER
3 REM *** FILES, OR TO REMOVE LAST END OF FILE, OR TO COPY FILE
4 REM *** INTO LARGER FILE
10 CLEAR 2000
15 REM *** TAKE GREAT PAINS IN CHECKING FILE EXISTENCE
20 PRINT "WHAT IS THE NAME OF THE EXISTING DATA FILE (WHICH MUST BE"
25 PRINT "ON THE DISKETTE IN DRIVE 1)";
40 FILE$="": INPUT FILE$
50 IF FILE$="" THEN 20
60 ON ERROR GOTO 90
70 OPENR FILE$+":1"
80 GOTO 130
90 RESUME 100
100 CLOSR
110 PRINT"ERROR--DATA FILE MISSING FROM DISKETTE ON DRIVE 1.  TRY AGAIN"
120 GOTO 20
130 PRINT "WHAT IS THE NAME OF THE DATA FILE YOU ARE ABOUT TO CREATE";
140 N$="": INPUT N$
150 IF N$="" THEN 130
160 PRINT "ON WHICH DRIVE IS THE DISKETTE WITH FILE ";N$;" (0 OR 1)";
170 I$="": INPUT I$
180 IF I$="0" THEN 200
190 IF I$<>"1" THEN 160
200 ON ERROR GOTO 230
210 OPENW N$+":"+I$
220 GOTO 280
230 RESUME 240
240 PRINT "FILE ";N$;" IS NOT ON THE DISKETTE IN DRIVE ";I$
250 PRINT "PLEASE FIX THE PROBLEM AND RETYPE THE PROPER VALUES."
260 CLOSW
270 GOTO 130
280 PRINT "TYPE 1 TO COPY ENTIRE FILE, 2 TO COPY A PORTION OF THE FILE,"
290 PRINT "OR 0 TO QUIT THE PROGRAM ENTIRELY AND NOT COPY ANYTHING."
300 C=-1: INPUT C
310 IF C=0 THEN 9000
320 IF C<1 OR C>2 THEN 280
330 IF C=2 THEN 410
340 ON ERROR GOTO 4000
350 DSKIN H
360 DSKOUT H
365 ON ERROR GOTO 400
370 A$="": DSKIN A$
380 DSKOUT A$
390 GOTO 365
400 RESUME 9000
410 PRINT "TYPE IN AN ENTRY NAME--IT NEED NOT EXIST IN FILE ";FILE$
420 PRINT "THE DATA FOR ANY ENTRY WITH NAME ALPHABETICALLY"
425 PRINT "GREATER THAN THE NAME YOU TYPE IN WILL BE PLACED IN FILE "
430 PRINT N$;".  (TYPE JUST THE RETURN KEY TO QUIT THE PROGRAM.)"
```



```
450 A$="": INPUT A$
460 IF A$="" THEN 9000
470 ON ERROR GOTO 4000
475 REM *** SAVE FILE HEADING INFORMATION
480 DSKIN H
490 DSKOUT H
500 FOR I=1 TO H
510 B$="": DSKIN B$
520 DSKOUT B$
530 NEXT I
540 ON ERROR GOTO 680
550 B$="": DSKIN B$
555 REM *** MAKE COMPARISON WITH USER-SPECIFIED NAME
560 IF B$>A$ THEN 610
565 IF H=1 THEN 540  'MAKE SURE NEXT LOOP ISN'T EXECUTED
566 REM *** IGNORE REST OF DATA FOR STUDENT
570 FOR I=1 TO H-1
580 DSKIN B$
590 NEXT I
600 GOTO 540
605 REM *** SAVE DATA, STARTING WITH NAME (ALREADY READ IN FROM FILE)
610 DSKOUT B$
620 IF H=1 THEN 540  'BE CAREFUL IF FILE ONLY CONTAINS NAMES
630 FOR I=1 TO H-1
640 B$="": DSKIN B$
650 DSKOUT B$
660 NEXT I
670 GOTO 540
680 RESUME 690
690 PRINT "SHOULD THE DATA YOU JUST STORED IN FILE ";N$
700 PRINT "BE REMOVED FROM FILE ";FILE$;" (Y OR N)";
720 B$="": INPUT B$
730 IF LEFT$(B$,1)="N" THEN 9000
740 IF LEFT$(B$,1)<>"Y" THEN 690
745 REM *** PLAY AROUND WITH FILES AGAIN; BE CAREFUL EVERYTHING IS OK
750 CLOSR
760 CLOSW
770 ON ERROR GOTO 800
780 OPENW "SCR:0"
790 GOTO 860
800 RESUME 810
810 CLOSW
820 PRINT "PLACE A DISKETTE LABELLED 'SCRATCH' IN DRIVE 0."
830 PRINT "THEN PRESS THE RETURN KEY ON THE TERMINAL."
840 INPUT B$
850 GOTO 770
860 ON ERROR GOTO 890
870 OPENR FILE$+":1"
880 GOTO 940
890 RESUME 900
```

SPLIT CONTINUED

```
900 PRINT "LEAVE THE DISKETTE WITH FILE ";FILE$;" IN DRIVE 1!!"
910 PRINT "PRESS THE RETURN KEY ON THE TERMINAL WHEN YOU'RE READY."
915 INPUT B$
920 CLOSR
930 GOTO 860
940 ON ERROR GOTO 4000
945 REM *** STORE HEADING INFO AGAIN
950 DSKIN H
960 DSKOUT H
970 FOR I=1 TO H
980 B$="": DSKIN B$
990 DSKOUT B$
1000 NEXT I
1005 REM *** NOW LOOK FOR ALL ITEMS NOT PLACED IN NEW FILE
1010 ON ERROR GOTO 1100
1020 B$="": DSKIN E$
1030 IF B$>A$ THEN 1110  'FOUND FIRST ITEM STORED IN NEW FILE YET?
1035 REM *** STORE ITEM IN OLD FILE
1040 DSKOUT B$
1045 IF H=1 THEN 1010  'BE CAREFUL NOT TO EXECUTE NEXT LOOP
1046 REM *** STORE REST OF DATA (NAME ALREADY READ IN FROM FILE)
1050 FOR I=1 TO H-1
1060 B$="": DSKIN E$
1070 DSKOUT B$
1080 NEXT I
1090 GOTO 1010  'GET NEXT NAME
1100 RESUME 1110
1105 REM *** TIME TO COPY FROM DUMMY FILE BACK INTO ORIGINAL
1110 CLOSR
1120 CLOSW
1125 ON ERROR GOTO 4000
1130 OPENW FILE$+" 1"
1140 OPENR "SCR:0"
1150 DSKIN H
1160 DSKOUT H
1170 ON ERROR GOTO 1210
1180 B$="": DSKIN E$
1190 DSKOUT B$
1200 GOTO 1170
1210 RESUME 9000
3999 REM *** UNEXPECTED ERROR IN DISK READS, FIND OUT WHERE HAPPENED
4000 PRINT "PLEASE WRITE DOWN THE FOLLOWING INFO AND TURN IT IN."
4005 PRINT "ALSO WRITE DOWN WHAT YOU DID (AS WELL AS YOU CAN REMEMBER)."
4006 PRINT "A MISTAKE HAPPENED THAT CANNOT BE FIXED RIGHT NOW."
4010 PRINT "ERROR #";ERR;"IN LINE #";ERL
4020 STOP
8999 REM *** NORMAL EXITING PROCEDURE
9000 CLOSR
9010 CLOSW
9015 REM *** MAKE SURE FDOS DISKETTE IS IN DRIVE 0
9020 ON ERROR GOTO 9060
9030 OPENR "EXEC"
9040 CLOSR
9045 PRINT "ALL FINISHED!!"
9050 GOTO 9999
9060 RESUME 9070
9070 CLOSR
9080 PRINT "PLACE A DISKETTE LABELLED 'PROGRAMS' IN DRIVE 0."
9090 PRINT "THEN PRESS THE RETURN KEY ON THE TERMINAL."
9100 INPUT B$
9110 GOTO 9020
9999 END
```

EXPAN

This program is not really needed if SPLIT is implemented for your system in its present form. It's only function is to copy a data file into a newly created file of larger size.

The following list contains all the variables used in EXPAN and the function each variable serves.

Name	Description
E$	name of the old data file
E1$	used to input the drive number in which the old file is located
F$	name of the new data file
F1$	used to input the drive number in which the new file is located
N	used to input the number of headings in the old data file
N$	used to input character data from the old file

EXPAN LISTING

```
10 CLEAR 5000 'RESERVE STRING-HANDLING SPACE
20 PRINT "NAME OF FILE YOU WANT TO EXPAND";
30 E$="": INPUT E$
40 IF E$="" THEN 20
50 PRINT "WHICH DRIVE IS ";E$; " ON (0 OR 1)";
60 E1$="": INPUT E1$
70 IF E1$="0" OR E1$="1" THEN 80 ELSE 50
80 ON ERROR GOTO 105
90 OPENR E$+":"+E1$
100 GOTO 150
105 RESUME 110
110 PRINT "ERROR*** ";E$;" DOES NOT EXIST ON DRIVE ";E1$
120 PRINT "PLEASE TRY AGAIN"
130 CLOSR
140 GOTO 20
150 PRINT "NAME OF LARGER FILE";
160 F$="": INPUT F$
170 IF F$="" THEN 150
180 PRINT "WHICH DRIVE IS ";F$; " ON (0 OR 1)";
190 F1$="": INPUT F1$
200 IF F1$="0" OR F1$="1" THEN 210 ELSE 180
210 ON ERROR GOTO 240
220 OPENW F$+":"+F1$
230 GOTO 280
240 RESUME 245
245 PRINT "ERROR*** ";F$;" DOES NOT EXIST ON DRIVE ";F1$
250 PRINT "PLEASE TRY AGAIN "
260 CLOSW
270 GOTO 150
280 DSKIN N
290 DSKOUT N
300 ON ERROR GOTO 340
310 N$="": DSKIN N$
320 DSKOUT N*
330 GOTO 300
340 RESUME 350
350 CLOSR
360 CLOSW
370 PRINT "ALL FINISHED"
380 END
```

COMB

This program spends a great deal of time checking the heading names used in the two files being merged to be sure that the first heading has the same name in both files and to compare the other headings so the output file will include all possible data categories. The program could be somewhat shortened by leaving out some of these checks and permitting only similar files (those which have exactly the same heading names in the same order) to be merged.

The following list contains all the variables used in COMB and the function each variable serves.

Name	Description
A$	used to input data from files
B$	stores the last entry name not yet entered in the add-on file
C$	stores a blank line for missing data
F	counts the actual number of headings in the files
F1(H1+H2)	flags to indicate what order in which to output data
F2(H1+H2)	flags to indicate what order in which to output data (negative value indicates a blank field)
F5	flag to indicate that the same name does not exist in both data files
F6	counts the number of headings in the oldest file which are not also in the add-on file
H1	number of headings in the oldest file
H1$(H1)	names of the headings in the oldest file
H2	number of headings in the add-on file
H2$(H2)	names of the headings in the add-on file
I	loop parameter
J	loop parameter
N$	name of the new file to be created
N1$	number of the drive in which the new file is located
P$	name of the oldest file
P1$	number of the drive in which the oldest file is located
S$	name of the add-on file
S1$	number of the drive in which the add-on file is located
T	used to input numbers from the data files
T$	used to input character data from the data files

COB LISTING

```
1 REM *** PROGRAMMED BY VIK MUIZNIEKS 1/78.  THIS PROGRAM ALLOWS THE
2 REM *** USER TO MERGE 2 FILES INTO 1.  THE ORIGINAL FILES ARE UNALTERED
3 REM *** THE FILES MUST HAVE THE SAME FIRST HEADING NAME, BUT THEY CAN
4 REM *** HAVE DIFFERENT NUMBERS OF HEADINGS, AND DIFFERENT HEADING
5 REM *** NAMES.  DATA FROM THE ADD-ON FILE OVERWRITES DATA FROM THE
6 REM *** OLDEST FILE IF THE NAMES ARE THE SAME IN BOTH FILES.
10 CLEAR 2000   'RESERVE STRING STORAGE SPACE
15 C$="":  'BLANK FOR MISSING FIELD
18 REM *** GET ALL FILE NAMES, MAKE SURE THEY EXIST BY OPENING
20 PRINT "NAME OF 'OLDEST' FILE";
30 P$="": INPUT P$
40 IF P$="" THEN 20
50 PRINT "ON WHICH DRIVE IS THE DISKETTE WITH FILE ";P$;" (0 OR 1)";
60 P1$="": INPUT P1$
70 IF P1$="0" THEN 81
80 IF P1$<>"1" THEN 50
81 ON ERROR GOTO 84
82 OPENR P$+":"+P1$
83 GOTO 88
84 RESUME 85
85 PRINT "ERROR--FILE ";P$;" NOT ON DRIVE ";P1$;".  TRY AGAIN PLEASE."
86 CLOSR
87 GOTO 20
88 P$=P$+":"+P1$
89 CLOSR
90 PRINT "NAME OF 'ADD-ON' FILE";
100 S$="": INPUT S$
105 IF S$="" THEN 90
110 PRINT "ON WHICH DRIVE IS THE DISKETTE WITH FILE ";S$;" (0 OR 1)";
120 S1$="": INPUT S1$
130 IF S1$="0" THEN 141
140 IF S1$<>"1" THEN 110
141 ON ERROR GOTO 144
142 OPENR S$+":"+S1$
143 GOTO 148
144 RESUME 145
145 CLOSR
146 PRINT "ERROR--FILE ";S$;" NOT IN DRIVE ";S1$;".  PLEASE TRY AGAIN."
147 GOTO 90
148 S$=S$+":"+S1$
150 PRINT "NAME OF NEW FILE TO BE CREATED";
160 N$="": INPUT N$
170 IF N$="" THEN 150
180 PRINT "ON WHICH DRIVE IS THE DISKETTE WITH FILE ";N$;" (0 OR 1)";
190 N1$="": INPUT N1$
200 IF N1$="0" THEN 220
210 IF N1$<>"1" THEN 180
220 ON ERROR GOTO 250
230 OPENW N$+":"+N1$
240 GOTO 300
250 RESUME 260
260 CLOSW
270 PRINT "ERROR--FILE ";N$;" NOT ON DRIVE ";N1$;".  PLEASE TRY AGAIN."
290 GOTO 150
295 REM *** FILES OPENED, GET HEADING INFO FROM EXISTING FILES
300 ON ERROR GOTO 1370  'NO ERRORS SHOULD OCCCUR YET
305 PRINT "THIS IS PROBABLY GOING TO TAKE A WHILE.  I SUGGEST YOU GO"
306 PRINT "DO SOMETHING ELSE AND COME BACK LATER TO SEE IF I'M DONE."
310 DSKIN H2 '# OF HEADINGS IN ADD-ON FILE
320 DIM H2$(H2)  'HEADING NAMES
330 FOR I=1 TO H2
340 H2$(I)="": DSKIN H2$(I)  'GET HEADING NAMES
350 NEXT I
360 CLOSR
370 OPENR P$
380 DSKIN H1  'DO SAME FOR OLDEST FILE
390 DIM H1$(H1)
400 FOR I=1 TO H1
410 H1$(I)="": DSKIN H1$(I)
420 NEXT I
425 IF H1$(1)<>H2$(1) THEN 1520  '1ST HEADING NAMES MUST BE THE SAME
430 DIM F1(H1+H2),F2(H1+H2) 'MAX # OF DIFFERENT HEADING NAMES
431 REM F1,F2 FLAG WHICH ORDER TO PRINT HEADINGS FROM RESPECTIVE FILES
432 REM IF NEG VALUE, THEN SHOULD OUTPUT A BLANK SINCE HEADING NOT AVAIL
435 F=H1+1  'COUNTS ACTUAL # OF DIFFERENT HEADINGS
436 F6=0  'COUNTS # OF ELEMENTS H2 HAS LESS THAN H1
440 FOR I=1 TO H1
441 F1(I)=I  'HEADING NAMES START WITH THOSE IN OLDEST FILE
442 IF I<=H2 THEN 450  'LESS HEADINGS IN ADD-ON FILE?
443 FOR J=1 TO H2
444 IF H1$(I)<>H2$(J) THEN 447 'EXISTS AS DIFFERENT NUMBER?
445 F2(J)=I
446 GOTO 560
447 NEXT J
448 F2(H2+F-H1+F6)=-I: F6=F6+1  'ADD ON ANOTHER HEADING TO ADD-ON
449 GOTO 560
450 IF H1$(I)<>H2$(I) THEN 490  'DIFFERENT HEADING NAMES?
470 F2(I)=I  'SAME ORDER
480 GOTO 560
490 FOR J=1 TO H2 'SEE IF HEADING SAME BUT IN DIFFERENT PLACE
500 IF H1$(I)<>H2$(J) THEN 530 'YES?
510 F2(J)=I  'STORE APP. ORDER #
520 GOTO 560
530 NEXT J
```

```
CONTINUED
535 F2(H2+F-H1+F6)=-I 'STORE OLDEST HEADING NOT PRESENT IN ADD-ON FILE
540 F2(I)=F  'ASSIGN NEXT NEW HEADING NUMBER
545 F1(F)=-F  'STORE OLDEST FILE DOES NOT HAVE HEADING FROM ADD-ON FILE
550 F=F+1  'INCREASE COUNTER ON # OF DIFFERENT HEADINGS
560 NEXT I
565 IF H1>=H2 THEN 630  'ALL HEADINGS FOUND?
570 FOR J=H1+1 TO H2  'FOR REMAINDER OF HEADINGS
575 FOR I=1 TO H2  'LOOK AT ALL HEADINGS
580 IF F2(I)=0 THEN 590  'FOUND ONE WITHOUT A #
585 NEXT I
586 PRINT "AN ERROR HAS OCCURRED THAT CANNOT BE FIXED."
587 PRINT "PLEASE TELL THE PROGRAMMER WHAT HAPPENED."
588 GOTO 1550
590 F2(I)=F  'GIVE NEXT HEADING # IN ADD-ON FILE
600 F1(F)=-F  'INDICATE NOT PRESENT IN OLDEST FILE
610 F=F+1  'INCREASE COUNTER
620 NEXT J
630 F=F-1  'ADJUST COUNTER TO PROPER NUMBER
635 REM BEGIN OUTPUT FILE IN PROPER FORMAT
640 DSKOUT F  '# OF HEADINGS IN NEW FILE
650 FOR I=1 TO H1
660 DSKOUT H1$(I)  'HEADING NAMES FOR NEW FILE
670 NEXT I
680 IF F=H1 THEN 760 'ALL HEADING NAMES OUTPUT?
690 FOR I=H1+1 TO F  'OUTPUT DIFFERENT NAMES FROM ADD-ON FILE
700 FOR J=1 TO H2
710 IF F2(J)<>I THEN 740 'FOUND HEADING?
720 DSKOUT H2$(J)
730 GOTO 750
740 NEXT J
745 PRINT "AN ERROR HAS OCCURRED."
746 GOTO 1400  'INFORM USER OF PROBLEM
750 NEXT I
755 REM NOW BEGIN COPYING IN DATA
760 ON ERROR GOTO 900 'COPY IN ADD-ON FILE SINCE OLDEST EXHAUSTED
763 A$="": DSKIN A$ 'A$ HOLDS LAST NON-ENTERED NAME IN OLDEST FILE
765 CLOSR
770 ON ERROR GOTO 970  'COPY IN OLDEST FILE, ADD-ON EXHAUSTED
773 F5=0  'SAME NAME NOT FOUND IN BOTH FILES
775 OPENR S$
776 DSKIN H2  'REREAD HEADING INFO
777 FOR I=1 TO H2
778 DSKIN B$
779 NEXT I
780 B$="": DSKIN B$ 'B$ HOLDS LAST NON-ENTERED NAME IN ADD-ON FILE
785 IF B$=A$ THEN F5=1  'FLAG EQUAL NAMES SO A$ NOT REENTERED
790 IF B$<=A$ THEN GOSUB 1050 ELSE 810  'COPY IN DATA?
794 ON ERROR GOTO 970 'CHECK ON FILE EXHAUSTION
795 GOTO 780  'GET NEXT NAME
810 CLOSR
815 ON ERROR GOTO 1370 'UNEXPECTED ERROR
820 OPENR P$
830 GOSUB 1170  'READ BACK UP TO A$
831 IF F5<>1 THEN 840  'SEE IF DUPLICATE NAME
832 FOR I=1 TO H1-1 'SKIP REST OF DATA
833 DSKIN A$
834 NEXT I
835 F5=0  'RESET FLAG BACK
836 GOTO 844 'GO ON AS IF COPIED IN
840 IF A$<B$ THEN GOSUB 1270 ELSE 850  'NEED TO COPY INTO FILE?
844 ON ERROR GOTO 900 'CHECK ON FILE EXHAUSTION
845 A$="": DSKIN A$  'GET NEXT NAME
846 GOTO 840 'AND KEEP ON PROCESSING
850 IF A$<>B$ THEN 860 'SEE IF DUPLICATE NAMES
851 FOR I=1 TO H1-1 'IF SO, SKIP ITEM
852 DSKIN A$
853 NEXT I
854 ON ERROR GOTO 900 'MAKE SURE RECOVER ON EMPTY FILE
855 A$="": DSKIN A$
860 CLOSR
865 ON ERROR GOTO 1370  'CHECK ON EMPTY ADD-ON FILE
870 OPENR S$
880 GOSUB 1420 'POSITION TO START OF OLD NAME IN ADD-ON FILE
890 GOTO 785  'AND PROCESS AS BEFORE
895 REM OLDEST FILE EXHAUSTED, COPY IN REMAINDER OF ADD-ON FILE
900 IF ERR=27 THEN RESUME 910 ELSE 1370  'CHECK ON ERROR CONDITION
910 CLOSR
920 OPENR S$
930 GOSUB 1420 'POSITION TO PROPER PLACE
950 GOSUB 1050 'ADD DATA FOR NAME
954 ON ERROR GOTO 9000 'ALL DONE
955 B$="": DSKIN B$ 'GET NEXT NAME
960 GOTO 950 'AND STORE IT
965 REM ADD-ON FILE EXHAUSTED, COPY IN REMAINDER OF OLDEST FILE
970 IF ERR=27 THEN RESUME 980 ELSE 1370 'CHECK ON ERROR
980 CLOSR
990 REM SAME COMMENTS APPLY AS ABOVE ROUTINE
1000 OPENR P$
1010 GOSUB 1170
1020 GOSUB 1270
1025 ON ERROR GOTO 9000
1030 A$="": DSKIN A$
1040 GOTO 1020
1045 REM COPY DATA VALUES FROM ADD-ON FILE INTO NEW FILE
1050 DSKOUT B$ 'SAVE NAME
1055 IF H2=1 THEN 1085  'NO MORE DATA ON NAME?
1060 FOR I=2 TO H2 'GET REST OF DATA VALUES
1070 H2$(I)="": DSKIN H2$(I)
1080 NEXT I
```

CDFB CONTINUED

```
1085 IF F=1 THEN RETURN 'ALL DONE?
1090 FOR I=2 TO F  'STORE REST OF HEADINGS
1095 FOR J=2 TO F 'LOOK FOR ALL POSSIBLE HEADINGS
1100 IF ABS(F2(J))<>I THEN 1150  'FOUND NEXT HEADING?
1110 IF F2(J)<0 THEN 1140  'SET TO BLANK?
1120 DSKOUT H2$(J)  'OUTPUT DATA FROM FILE
1130 GOTO 1155
1140 DSKOUT C$  'OUTPUT BLANK'
1145 GOTO 1155
1150 NEXT J
1151 PRINT "ERROR  ";
1152 GOTO 1400  'STOP PROGRAM AFTER INFORMING USER OF PROBLEM
1155 NEXT I
1160 RETURN
1165 REM READ UP TO PREVIOUS NAME IN OLDEST FILE
1170 DSKIN T  'START BY SKIPPING HEADING INFORMATION
1180 FOR I=1 TO T
1190 DSKIN T$
1200 NEXT I
1210 T$="": DSKIN T$ 'GET NEXT NAME
1220 IF T$=A$ THEN RETURN 'FOUND THE ONE THAT'S BEING LOOKED FOR?
1225 IF T=1 THEN 1210  'ONLY ONE HEADING IN FILE?
1230 FOR I=1 TO T-1  'SKIP REST OF DATA ON THIS ENTRY
1240 DSKIN T$
1250 NEXT I
1260 GOTO 1210  'AND GET NEXT NAME
1265 REM COPY DATA FOR ONE ENTRY FROM OLDEST FILE
1270 DSKOUT A$  'STORE NAME
1275 IF H1=1 THEN 1320 'SEE IF NO MORE DATA IN OLDEST FILE
1280 FOR I=1 TO H1-1  'GET REST OF DATA
1290 T$="": DSKIN T$
1300 DSKOUT T$
1310 NEXT I
1320 IF H1=F THEN RETURN 'ALL DONE?
1330 FOR I=H1+1 TO F 'OUPUT BLANKS FOR REST OF DATA
1340 DSKOUT C$
1350 NEXT I
1360 RETURN
1365 REM ERROR ROUTINE; UNRECOVERABLE ERROR OCCURRED
1370 PRINT "ERROR #";ERR;"IN LINE #";ERL
1375 RESUME 1380
1380 PRINT "AN ERROR HAS OCCURRED IN THE PROGRAM.  PLEASE WRITE DOWN"
1390 PRINT "AS MUCH AS YOU CAN ABOUT WHAT YOU DID AND TURN IT IN."
1400 PRINT "THE NEW FILE HAS NOT BEEN CORRECTLY CREATED."
1402 CLOSW
```



```
1405 CLOSR
1410 STOP
1415 REM READ UP TO PREVIOUS VALUE IN ADD-ON FILE
1420 DSKIN T  'START BY REREADING HEADING INFO
1430 FOR I=1 TO T
1440 DSKIN T$
1450 NEXT I
1460 T$="": DSKIN T$  'GET NEXT NAME
1470 IF T$=B$ THEN RETURN 'FOUND NAME THAT'S BEING LOOKED FOR?
1475 IF T=1 THEN 1460 'ONLY NAMES IN FILE
1480 FOR I=1 TO T-1 'IF NOT, SKIP REST OF DATA
1490 DSKIN T$
1500 NEXT I
1510 GOTO 1460  'GO GET NEXT NAME
1515 REM INFORM USER THAT FILES ARE NOT COMPATIBLE.
1520 PRINT "THE FILES DO NOT HAVE THE SAME FIRST HEADING NAME."
1530 PRINT "THEREFORE, I CANNOT MERGE THE TWO FILES TOGETHER."
1540 PRINT "SORRY, BUT YOU'LL HAVE TO TRY MERGING WITH DIFFERENT FILES."
1550 CLOSW
1560 CLOSR
1570 STOP
8999 REM NORMAL EXITING ROUTINE
9000 CLOSR  'CLOSE FILE CHANNELS
9010 CLOSW
9015 REM INFORM USER THAT MERGE IS FINISHED.
9020 PRINT N$;" HAS BEEN CREATED."
9025 REM MAKE SURE FDOS DISKETTE IN DRIVE 0.
9030 ON ERROR GOTO 9060
9040 OPENR "EXEC"
9045 CLOSR
9050 GOTO 9110 'ALL DONE
9060 RESUME 9070
9070 CLOSR
9075 PRINT "PLEASE PLACE THE DISKETTE LABELLED 'PROGRAMS' IN DRIVE 0."
9080 PRINT "WHEN READY PRESS THE RETURN KEY ON THE TERMINAL."
9090 T$="": INPUT T$
9100 GOTO 9030
9110 END
```

CHANG

This program could be split into several smaller programs each with one specific function if memory space is limited. The procedure of entering new or altered data for each entry (lines 900-920 and 1490-1510) is time-consuming and probably unnecessary since the INFO program can be used for this purpose. That option, therefore, could be omitted.

The following list contains all the variables used in CHANG and the function each variable serves.

Name	Description
A$	name of the new heading
B$	used to input data from the file
C	used to input responses to system messages for routing
C1	used to input the number of the heading which a new heading should follow
C$	saves the standard to be stored under each entry, if needed
CASE	flag used to indicate if starting the program a second time
FILE$	name of the data file (must be in drive 1)
HDNAME$(HNUM)	names of headings in the data file
HNUM	number of headings in the data file
SCR$	name of scratch file (SCR) in drive 0

OWNS LISTING

```
10 REM *** THIS PROGRAM ALLOWS USER TO ADD, DELETE, OR CHANGE HEADING
20 REM *** NAMES.  IF ADDING OR CHANGING, USER CAN SELECT FROM SEVERAL
30 REM *** OPTIONS FOR MANIPULATING/INITIALIZING DATA UNDER HEADING.
40 REM *** PROGRAMMED BY VIK MUIZNIEKS 11/77
50 REM ***
60 PRINT "ALWAYS PRESS THE RETURN KEY AFTER TYPING-IN A RESPONSE!!!!!"
70 CLEAR 2000  'SET UP STRING STORAGE SPACE
80 PRINT "MAKE SURE A DISKETTE LABELLED 'SCRATCH' IS IN DRIVE 0."
90 PRINT "NAME OF THE DATA FILE"
100 PRINT "THE FILE MUST BE ON A DISKETTE IN DRIVE 1."
110 FILE$="": INPUT FILE$
120 IF FILE$="" THEN 90
130 FILE$=FILE$+":1"
140 SCR$="SCR:0"
150 CASE=0  'NO REINITIALIZATION
160 IF CASE=0 THEN 200
170 CLOSR
180 CLOSW
190 ERASE HDNAME$
200 ON ERROR GOTO 280
210 OPENR FILE$
220 ON ERROR GOTO 250
230 OPENW SCR$
240 GOTO 320
250 RESUME 260
260 CLOSW
270 GOTO 290
280 RESUME 290
290 CLOSR
300 PRINT "***ERROR***--FILES NOT ON PROPER DISKETTES!"
310 GOTO 80
320 HNUM=0:DSKIN HNUM  'READ NUMBER OF HEADINGS
330 DIM HDNAME$(HNUM)
340 FOR I=1 TO HNUM  'READ HEADING NAMES
350 HDNAME$(I)="":DSKIN HDNAME$(I)
360 NEXT I
370 PRINT
380 PRINT "CHOICES:   1) ADD A NEW HEADING"
390 PRINT "           2) CHANGE A HEADING NAME"
400 PRINT "              (AND/OR DATA UNDER HEADING)"
410 PRINT "           3) DELETE A HEADING"
420 PRINT "           4) LEAVE THIS PROGRAM"
430 PRINT " TYPE IN THE NUMBER OF YOUR CHOICE";
440 C=0:INPUT C
450 IF C<1 OR C>4 THEN 370
460 ON C GOTO 480,1070,1580,2020
470 REM *** ADD A NEW HEADING
480 PRINT"TYPE 1 TO ENTER IN ALL THE DATA UNDER THE NEW HEADING NOW, OR"
490 PRINT "TYPE 2 TO HAVE ALL THE DATA UNDER THE HEADING SET BLANK, OR"
500 PRINT "TYPE 3 TO SET ALL THE DATA UNDER THE HEADING TO A VALUE YOU";
510 PRINT "CHOOSE.";
520 C=0:INPUT C
530 IF C<1 OR C>3 THEN 480
540 PRINT "HERE ARE THE CURRENT HEADING NAMES AND THEIR NUMBERS:"
550 FOR I=1 TO HNUM
560 PRINT I;") ";HDNAME$(I)
570 NEXT I
580 PRINT "TYPE IN THE NUMBER AFTER WHICH YOU WANT TO INSERT THE NEW"
590 PRINT "HEADING (OR TYPE 0 NOT TO ADD A NEW HEADING)";
600 REM *** CAN'T INSERT AS FIRST HEADING BECAUSE ALPHABETIZED ALREADY
610 C1=-1:INPUT C1
620 IF C1=0 THEN 370
630 IF C1<1 OR C1>HNUM THEN 580
640 PRINT "WHAT DO YOU WANT TO CALL THE NEW HEADING";
650 A$="": INPUT A$
660 IF A$<>"" THEN 690
670 PRINT "YOU MUST TYPE IN SOMETHING!"
680 GOTO 640
690 DSKOUT HNUM+1 'INCREMENT NUMBER OF HEADINGS
700 IF C<>3 THEN 730  'GET DATA VALUE IF NECESSARY
710 PRINT "WHAT IS THE VALUE YOU WANT ALL THE DATA TO BE SET TO";
720 C$="": INPUT C$
730 FOR I=1 TO C1  'THIS LOOP EXECUTED AT LEAST ONCE
740 DSKOUT HDNAME$(I)  'ADD HEADING NAME TO FILE
750 NEXT I
760 DSKOUT A$  'ADD NEW HEADING NAME
770 IF C1=HNUM THEN 810  'SEE IF ALL HEADINGS IN
780 FOR I=C1+1 TO HNUM  'IF NOT, ADD REMAINDER
790 DSKOUT HDNAME$(I)
800 NEXT I
810 ON ERROR GOTO 1030
820 FOR I=1 TO C1 'COPY DATA VALUES
830 B$="":DSKIN B$
840 DSKOUT B$
850 IF C=3 THEN 870  'DON'T CHANGE C$ IF USING IT AS DATA VALUE
860 IF I=1 THEN C$=B$  'SAVE NAME OF ITEM
870 NEXT I
880 IF C=2 THEN 990  'SEE IF SET NEW DATA VALUE TO BLANK
890 IF C=3 THEN 1010 'SET DATA TO USER-SPECIFIED VALUE?
900 PRINT "VALUE OF ";A$;" FOR ";C$
910 B$="":INPUT B$
920 DSKOUT B$
930 IF C1=HNUM THEN 810  'SEE IF ALL COPIED
940 FOR I=C1+1 TO HNUM 'FINISH OFF ITEM
```

```
950 B$="":DSKIN B$
960 DSKOUT B$
970 NEXT I
980 GOTO 810
990 B$=""
1000 GOTO 920  'DATA ITEM SET TO BLANK
1010 B$=C$  'DATA ITEM SET TO SPECIFIED VALUE
1020 GOTO 920
1030 RESUME 1040
1040 GOSUB 1870 'COPY FILE OVER
1050 GOTO 190  'REINITIALIZE DATA FILE
1060 REM *** CHANGE NAME OF HEADING (AND POSSIBLY DATA)
1070 PRINT "TYPE 1 TO CHANGE DATA UNDER HEADING AS WELL, OR"
1080 PRINT "TYPE 2 TO LEAVE DATA UNDER HEADING ALONE, OR"
1090 PRINT "TYPE 3 TO CHANGE DATA TO A VALUE YOU SPECIFY. ";
1100 C1=0:INPUT C1
1110 IF C1<1 OR C1>3 THEN 1070
1120 IF C1<>3 THEN 1150
1130 PRINT "WHAT DO YOU WANT TO CHANGE THE DATA TO";
1140 C$="":  INPUT C$
1150 PRINT "HERE ARE THE CURRENT HEADINGS AND THEIR NUMBERS:"
1160 FOR I=1 TO HNUM
1170 PRINT I;") ";HDNAME$(I)
1180 NEXT I
1190 PRINT "TYPE THE NUMBER OF THE HEADING YOU WANT TO CHANGE, OR"
1200 PRINT "TYPE 0 NOT TO CHANGE ANY HEADING NAME.  REMEMBER, YOU"
1210 PRINT "CAN ONLY CHANGE THE HEADING NAME FOR THE FIRST HEADING"
1220 PRINT "(NOT THE DATA) SINCE THE DATA IS STORED ALPHABETICALLY."
1230 C=-1:INPUT C
1240 IF C=0 THEN 370  'RETURN TO MAIN CHOICES
1250 IF C=1 AND C1=1 THEN 1190  'CAN'T CHANGE FIRST HEADING DATA
1260 IF C=1 AND C1=3 THEN 1190
1270 IF C<1 OR C>HNUM THEN 1190
1280 PRINT "WHAT DO YOU WANT TO CHANGE ";HDNAME$(C);" TO?"
1290 A$="":  INPUT A$
1300 IF A$<>"" THEN 1330
1310 PRINT "YOU MUST TYPE IN SOMETHING!"
1320 GOTO 1280
1330 DSKOUT HNUM   'WRITE OUT NUMBER OF HEADINGS
1340 FOR I=1 TO HNUM   'WRITE OUT HEADING NAMES
1350 IF I=C THEN 1380   'CHANGED VALUE?
1360 DSKOUT HDNAME$(I)
1370 GOTO 1390
1380 DSKOUT A$
1390 NEXT I
1400 ON ERROR GOTO 1540
1410 FOR I=1 TO HNUM
1420 B$="":DSKIN B$
1430 IF C1=3 THEN 1460 'DON'T CHANGE VALUE OF C$ IF USING IT
1440 IF I=1 THEN C$=B$   'SAVE ENTRY NAME
1450 IF I=C AND C1=1 THEN 1490  'SEE IF NEED TO CHANGE DATA VALUE
1460 IF I=C AND C1=3 THEN B$=C$  'CHANGE TO DESIRED VALUE
1470 DSKOUT B$
1480 GOTO 1520
1490 PRINT "VALUE OF ";A$;" FOR ";C$
1500 B$="":INPUT B$
1510 GOTO 1470
1520 NEXT I
1530 GOTO 1400
1540 RESUME 1550
1550 GOSUB 1870 'COPY FILE OVER
1560 GOTO 190  'REINITIALIZE
1570 REM *** DELETE A HEADING
1580 PRINT "HERE ARE THE CURRENT HEADINGS AND THEIR NUMBERS:"
1590 FOR I=1 TO HNUM
1600 PRINT I;") ";HDNAME$(I)
1610 NEXT I
1620 PRINT "TYPE IN THE NUMBER OF THE HEADING YOU WANT TO DELETE, OR"
1630 PRINT "TYPE 0 NOT TO DELETE ANY HEADINGS.  HEADING NUMBER ONE"
1640 PRINT"CANNOT BE DELETED (SINCE FILE ALPHABETIZED UNDER HEADING 1)."
1650 C=-1:INPUT C
1660 IF C=0 THEN 370  'RETURN IF NO DELETION TO MAIN CHOICES
1670 IF C<2 OR C>HNUM THEN 1620
1680 PRINT "ARE YOU SURE YOU WANT TO DELETE ";HDNAME$(C);" (YES OR NO)";
1690 A$="":  INPUT A$
1700 IF LEFT$(A$,1)="N" THEN 370  'RETURN TO MAIN CHOICES
1710 IF LEFT$(A$,1)<>"Y" THEN 1680
1720 DSKOUT HNUM-1  'INDICATE ONE LESS HEADING IN FILE
1730 FOR I=1 TO HNUM
1740 IF I=C THEN 1760   'SKIP DELETED ENTRY
1750 DSKOUT HDNAME$(I)
1760 NEXT I
1770 ON ERROR GOTO 1840
1780 FOR I=1 TO HNUM  'COPY REST OF FILE
1790 A$="":DSKIN A$
1800 IF I=C THEN 1820  'DELETE DATA UNDER DELETED HEADING
1810 DSKOUT A$
1820 NEXT I
1830 GOTO 1770   'GET NEXT ENTRY
1840 RESUME 1850
1850 GOSUB 1870  'COPY FILE OVER
1860 GOTO 190   'REINITIALIZE FILES
1870 REM *** COPY SCRATCH FILE INTO DATA FILE
1880 CLOSE
1890 CLOSW
1900 OPENW FILE$
1910 OPENR SCR$
1920 HNUM=0:DSKIN HNUM
1930 DSKOUT HNUM
1940 ON ERROR GOTO 1980
```

CWNG CONTINUED

```
1950 A$="":DSKIN A$  'COPY ALL ITEMS, DON'T CARE WHAT THEY ARE
1960 DSKOUT A$
1970 GOTO 1940
1980 RESUME 1990
1990 CLOSR
2000 CLOSW
2010 RETURN  'DONE COPYING
2020 REM *** NORMAL ENDING SEQUENCE
2030 CLOSR
2040 CLOSW
2050 ON ERROR GOTO 2090  'MAKE SURE FDOS DISKETTE IN DRIVE 0
2060 OPENR "EXEC"
2070 CLOSR
2080 GOTO 2150
2090 RESUME 2100
2100 PRINT "PLACE THE DISKETTE LABELLED 'PROGRAMS' IN DRIVE 0."
2110 PRINT "THEN PRESS THE RETURN KEY ON THE TERMINAL."
2120 INPUT A$
2130 CLOSR
2140 GOTO 2050
2150 END
```

BIBLIOGRAPHY

The interested reader is referred to the following University publications for further information.

1. Gaede, Owen F. and Ted J. Singletary, "Computer Applications in Science Education", Illinois Series on Educational Application of Computers (ISEAC), No. 17e, Department of Secondary Education, University of Illinois, 1979

2. Hicks, Bruce, "MSEA-THE Microcomputer System for Educational Applications", ISEAC No. 20, Department of Secondary Education, University of Illinois, 1977

3. Muiznieks, Viktors, "MSEA-The Department's Microcomputer", ISEAC No. 27, Department of Secondary Education, University of Illinois, 1978

4. Muiznieks, Viktors J., Clara Symonds, and Mark W. Pelczarski, "INFOSH-A BASIC Program for File Search", ISEAC No. 13, Department of Secondary Education, University of Illinois, 1976

5. Pelczarsi, Mark, "INFO-A Data Manipulation and Retrieval System in BASIC", ISEAC No. 9, Department of Secondary Education, University of Illinois, 1976